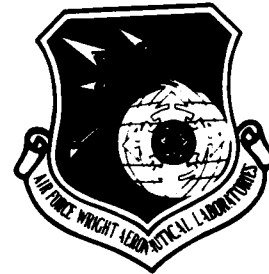


LEVEL II

2



AD A088066

AFWAL-TR-80-1046

Fault Tolerant Computer Network Study

International Business Machines Corporation
Federal Systems Division
Owego, NY 13827

DTIC
ECTE
AUG 20 1980
D
C

April 1980

Interim Report
September 1979 — March 1980

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433

DDC FILE COPY

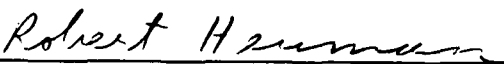
80 8 19 031

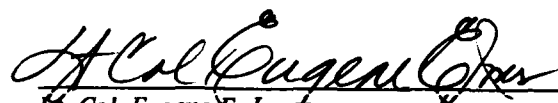
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

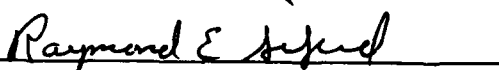
This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.


Robert Heuman
Project Engineer
System Design Group


Lt. Col. Eugene E. Jones
Chief
Avionic Systems Engineering Branch

FOR THE COMMANDER


Raymond E. Siferd, Colonel, USAF
Chief
System Avionics Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AAAA, W-PAFB, OH 45433 to help us maintain a current mailing list .

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFWAL/TR-88-1046	2. GOVT ACCESSION NO. AD A088066	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Fault Tolerant Computer Network Study		5. TYPE OF REPORT & PERIOD COVERED Interim Report Sept 79 - March 80
6. AUTHOR(s) T. Comfort J. Kaufman T. M. Lorman T. L. Anthony P. M. Kogge R. McNabb T. R. Battle A. W. Lord V. J. Smoral		7. PERFORMING ORG. REPORT NUMBER -88-D68-001
8. CONTRACT OR GRANT NUMBER(s) F33615-79-C-1180		9. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2003-01-17
10. CONTROLLING OFFICE NAME AND ADDRESS Avionics Laboratory Air Force Wright Aeronautical Laboratories Air Force Systems Command Wright-Patterson AFB, OH 45433		11. REPORT DATE April 1980
12. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 182-1		13. NUMBER OF PAGES 182
14. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE		
16. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
17. SUPPLEMENTARY NOTES		
18. KEY WORDS (Continue on reverse side if necessary and identify by block number) Fault Tolerance Distributed Processing Avionics Architecture Fault Tolerant Avionics		
19. ABSTRACT (Continue on reverse side if necessary and identify by block number) -This interim report documents some baseline parameters and considerations for a fault tolerant avionics system. Issues addressed include reliability, computational requirements, fault tolerant techniques, cost effectiveness, error types, computer networks, and analysis techniques. Candidate life-cycle cost models are reviewed for applicability in evaluating fault tolerant architectures, and a study plan for assessment of the cost/effectiveness of fault tolerant techniques is presented.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409206 Dm

PREFACE

This interim report documents the results of the 6-month Phase I portion of the Fault-Tolerant Computer Network Study, being performed by the Federal Systems Division of International Business Machines Corp., for the Air Force Avionics Laboratory. This report addresses specific tasks performed to meet the requirements of Air Force Contract F33615-79-C-1880.

This report was prepared under the technical leadership of Mr. Webb T. Comfort. Principal contributors to the report were Mr. T.L. Anthony, Mr. T.R. Battle, Mr. J.E. Kaufman, Dr. P.M. Kogge, Mr. A.W. Lord, Mr. T.M. Lorman, Mr. R.E. McNabb, and Mr. V.J. Smoral.

Mr. Robert Heuman, AFWAL/AAA-1 supervised the technical effort for the Air Force.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DDC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or special
A	

TABLE OF CONTENTS

Section		Page
1	Introduction	9
1.1	Summary of the Problem	9
1.2	Study Objectives	10
1.3	Methodology	10
2	Phase I Task Results	15
2.1	Task 1.1 - Computational Requirements	17
2.1.1	F-15A Avionic System Description	18
2.1.2	ATAS Functional Elements	20
2.1.2.1	Inertial Navigation	25
2.1.2.2	Air Data	25
2.1.2.3	General Navigation	25
2.1.2.4	Flight Director	26
2.1.2.5	Fire Control Radar	26
2.1.2.6	Air-to-Air Weapon Delivery	27
2.1.2.7	Air-to-Ground Weapon Delivery	27
2.1.2.8	Armament Control	27
2.1.2.9	Radar Warning	27
2.1.2.10	JTIDS	27
2.1.2.11	Controls and Displays Management	28
2.1.2.12	Display Signal Data Processing	28
2.1.3	Processing Requirements Summary	28
2.1.4	Candidate ATAS Fault Tolerant Computer Network Configurations	30
2.1.4.1	Distributed Central Computer Network (Candidate Number 1).	30
2.1.4.2	Widely Distributed Computer Network (Candidate Number 2) .	33
2.1.5	Functional Redundancy	33
2.1.5.1	Navigation and Guidance	36
2.1.5.2	Fixtaking	36
2.1.5.3	Communications and Identification	37
2.1.5.4	Tactical Electronic Warfare	37
2.1.5.5	Air-to-Ground Target Designation/Weapon Delivery	37
2.1.5.6	Air-to-Air Target Acquisition/Tracking/Weapon Delivery . .	37
2.2	Task 1.2 - Computer Reliability	38
2.3	Tasks 1.3 and 1.5 - Error Catalog	45
2.3.1	Definitions	46
2.3.2	Error Catalog	47
2.3.2.1	Memory Function	49
2.3.2.2	I/O Function	51
2.3.2.3	System Bus Function	52
2.3.2.4	CPU Function	53
2.4	Task 1.4 - Bus Control Procedures	55
2.4.1	Selection Approach	55

TABLE OF CONTENTS (Cont)

Section		Page
2.4.2	Selection Criteria	55
2.4.2.1	System Organization Criteria	56
2.4.2.2	Bus Access Requirements	57
2.4.2.3	Message Structure Requirements	58
2.4.2.4	Electrical and Technology Associated Criteria	58
2.4.3	Evaluation Data Buses	59
2.4.3.1	Stationary Master (MIL-STD-1553B) Data Bus	61
2.4.3.2	Nonstationary Master	62
2.4.3.3	Contention	63
2.4.3.4	Distributed, Tag-Encoded, Parallel, Selector Channel	64
2.4.3.5	Distributed, Tag-Encoded, Serial, Selector Channel	68
2.4.3.6	MIL-STD-1553B Data Bus with Overlay, Binary, Bit-for- Bit Priority	70
2.4.3.7	Overlapped Control, Parallel, High Performance, Distributed Bus	70
2.4.3.8	Overlapped Control, Serial, Distributed Bus	73
2.4.4	Candidate Bus Selection	74
2.5	Task 1.6 - Survey of Modeling Techniques	78
2.5.1	Performance Modeling	78
2.5.1.1	Survey of Performance Modeling Techniques for the Fault Tolerance Study	79
2.5.1.2	Fault Tolerance Performance Model Selection	84
2.5.1.2.1	Fault Tolerance Study Model Requirements	84
2.5.1.2.2	Performance Model Proposed for the Fault Tolerance Study ..	84
2.5.1.3	Fault Tolerance Model	86
2.5.1.3.1	Background	86
2.5.1.3.2	General Description	87
2.5.1.3.3	PROTIME II Features	88
2.5.1.3.3.1	User Interface	88
2.5.1.3.3.2	Input Data Preprocessor	89
2.5.1.3.3.3	Execution Model	92
2.5.1.3.4	Simulation Parameters	92
2.5.1.3.4.1	Simulation Control Data	93
2.5.1.3.4.2	Processor Descriptions	93
2.5.1.3.4.3	Device Descriptions	93
2.5.1.3.4.4	Event Description	94
2.5.1.3.4.5	Function Linkage Specifications	94
2.5.1.3.4.6	Function Requirements Description	95
2.5.1.3.5	Model Operation	95
2.5.1.3.5.1	Preprocessor Operation	95
2.5.1.3.5.2	Initialization and Static Parameters Calculation	97
2.5.1.3.5.3	Simulation Execution	97
2.5.1.3.5.4	Simulation Output	98

TABLE OF CONTENTS (Cont)

Section		Page
2.5.1.3.6	Output Data Description	98
2.5.1.3.6.1	Preprocessor Output	98
2.5.1.3.6.2	Input Data and Static Parameter Output	98
2.5.1.3.6.3	Simulation Statistics	99
2.5.1.3.6.4	Optional Snapshot Output	99
2.5.2	Reliability Modeling	100
2.5.2.1	Mathematical Model Fundamentals	101
2.5.2.2	Model Implementation	107
2.5.3	Life-Cycle Cost Modeling	109
2.5.3.1	Candidate LCC Models	109
2.5.3.1.1	Logistic Support Cost Model	109
2.5.3.1.2	Spares Optimized Inventory Level Selection (SOILS)	110
2.5.3.1.3	Life-Cycle Cost Model	110
2.5.3.2	Program Life-Cycle Cost	111
2.5.3.3	Analysis of Models/Selection of Application Model	112
2.5.3.4	Selected Model	113
2.5.3.4.1	Functional Sequence in Program LCC	113
2.5.3.4.2	Sequence to Operate Program LCC	115
2.5.3.4.3	Program LCC Input Parameter Identification/Definition	115
2.5.3.4.4	Program LCC Output Format	120
2.5.3.5	Summary/Conclusion	120
3	Scientific and Technical Information	123
3.1	Ultrasystems Study for AFAL	123
3.2	Unified Data System	124
3.3	Distributed Processor/Memory System	125
3.4	Wideband Multiplex Buses	126
3.5	Software Implemented Fault Tolerance	127
3.6	Ultrasystems Study for NASA	128
3.7	Draper Laboratories Study	129
3.8	DAIS Fault-Tolerance Study	130
3.9	Study for Commercial Aircraft Application	130
3.10	AASMMA Study	131
3.11	Carnegie-Mellon Studies in Fault Tolerance	132
3.12	MCF Study	133
3.13	Integrated Avionic System	134
4	Phase II Study Plan	137
4.1	Definition of the Problem	137
4.2	Outline of Technical Effort	141
4.2.1	Task 2.1 - Configuration Analysis and Selection	142
4.2.2	Task 2.2 - Fault Tolerance Analysis and Selection	143
4.2.3	Task 2.3 - Fault Tolerance Evaluation	144
4.2.4	Task 2.4 - Communication Protocol	144

TABLE OF CONTENTS (Cont)

Section		Page
4.2.4.1	Detailed Bus Analysis	145
4.2.4.2	Candidate Selection	146
4.2.4.3	Documentation	147
4.2.4.4	MIL-STD-1553B Evaluation	147
4.2.5	Task 2.5 - Conclusions and Recommended System	148
4.2.6	Task 2.6 - Advanced Technology Considerations	148
4.2.7	Task 2.7 - Final Report	149
4.3	Schedule	149
5	Summary of Key Results from Phase I Study	151
6	References	153
Appendix A	PROTIME II Model Simulation Methodology	155
A.1	Assumptions	155
A.2	Analytic Equations for Resource Consumption	155
A.3	Discrete Simulation Time Control	157
A.4	Analytic Equations for Resource Use	158
A.5	Model Execution	158
Appendix B	Sample PROTIME II Output	161
Appendix C	Sample Reports from Program LCC	171

LIST OF ILLUSTRATIONS

Figure		Page
1-1	General Methodology	11
1-2	Generic Distributed Computer Network	13
2-1	Representative F-15 Avionics Functional Block Diagram	19
2-2	F-15 Baseline Multicomputer Network	21
2-3	ATAS Data Processing Function Interfaces	23
2-4	Distributed Central Computer Network (Candidate Number 1) . .	31
2-5	Widely Distributed Computer Network (Candidate Number 2) . .	34
2-6	Distributed, Tag-Encoded, Selector Channel	65
2-7	Sample Overlay Priority Pole	67
2-8	Overlapped-Control, High Performance, Distributed Bus	72
2-9	Bus Comparison Breakdown	76
2-10	Overall Bus Comparison	77
2-11	Simulation Relationships	81
2-12	Sample Timeline - Single Processor System	84
2-13	Sample Page of Tutor	89
2-14	PROTIME II Model Data Entry Format	90
2-15	PROTIME II Data Entry Format	96
2-16	Program LCC Operational Flow	116
4-1	Error-Handling Procedure	138
4-2	Combined System Evaluation	141
4-3	Sections of a Bus Interface Unit	145
4-4	Phase II Schedule	149

LIST OF TABLES

Table		Page
2-1	Applicability of Phase I Tasks	15
2-2	ATAS Computational Requirements Summary	29
2-3	Distributed Computer Network Configuration 1 Requirements	32
2-4	Distributed Computer Network Configuration 2 Requirements	35
2-5	Processor Summary Chart	39
2-6	Typical Subassembly Failure Rates	40
2-7	Memory Page Failure Rates	40
2-8	Microprocessor and LSI Device Failure Rates	41
2-9	Power Supply Characteristics	41
2-10	Failure Rate Synthesis for AP-1	42
2-11	Failure Rate Synthesis for AP-101	42
2-12	Failure Rate Synthesis for AP-101C	43
2-13	Failure Rate Synthesis for AYK-14	43
2-14	Mission Failure Rate Synthesis for CC-1	44
2-15	Failure Rate Synthesis for CC-2	44
2-16	Selected Subsystem Reliability Information	45
2-17	Major Function Breakdown	46
2-18	Memory Function Error Catalog	49
2-19	Memory Function Error Checker Description	50
2-20	I/O Function Error Catalog	51
2-21	I/O Function Error Checker Descriptions	51
2-22	System Bus Function Error Catalog	52
2-23	System Bus Function Error Checker Description	52
2-24	CPU Function Error Catalog	53
2-25	CPU Function Error Checker Descriptions	54
2-26	Evaluation Data Buses	60
2-27	Data Bus Selection Matrix	75
2-28	Primary Roles of Software Simulation	83
2-29	Performance Modeling Approach Characteristics	85
2-30	Decision Criteria by Model	112
4-1	Evaluation Measures	140
4-2	Applicability of Phase II Tasks	142

Section 1 INTRODUCTION

1.1 SUMMARY OF THE PROBLEM

This program addresses the current high life-cycle costs of avionics systems. Because of their complexity, these systems have high design and development costs, initial integration costs, and initial test facilities costs. Once operational, avionics systems require many maintenance actions that must be performed by highly skilled personnel. In addition, current integrated digital avionics systems are not easily extended, necessitating very expensive upgrades when missions change or new threats are identified.

Current and projected computer technology offers the potential for significant reductions in avionics acquisition and support costs as well as improvements in weapon system availability, mission reliability, and extendability.

The application of distributed processing to the avionic computer subsystem suggests the possibility of cost savings in software development and support (through instruction set standardization) in addition to cost savings in hardware, maintenance, and spares. Improved diagnostic capability and fault-tolerant design appear to offer leverage in life-cycle cost as a result of the simplification of, and reduced requirements for, maintenance actions. The costs of adapting a platform for a given mission may also be reduced by these techniques.

Previous studies have addressed the application of a distributed network to real-time problems, and the engineering aspects of achieving fault tolerance within processor elements and within the processing network. Other studies have focused on the life-cycle cost benefits of standardization of hardware and software. This study addresses the selection of an optimum, cost-effective, fault-tolerant distributed avionics computer network design, and the methodology used in that selection.

IBM's interest in fault-tolerant, distributed processing, and life-cycle cost is long standing. As a system integrator for programs such as LAMPS and AN/BQQ-5, life-cycle cost effectiveness and attractive approaches to enhancing life-cycle cost effectiveness have been a major concern. The potential savings postulated for distributed processing have motivated significant research and development efforts as well as several notable implementations of distributed networks for communications and electronic warfare applications. While the component technology and the standardization benefits have readily evidenced themselves, the software costs, for both development and support, are less readily characterized.

The specific issue being addressed in this study is the cost effectiveness of fault tolerance. While the hardware cost adders associated with various fault tolerance techniques are easily quantified, the potential software development and support costs must also be understood.

Furthermore, savings in maintenance beyond those already accrued from improved diagnostics represent only the direct consequences of adding fault tolerance. The most significant savings to be derived from fault tolerant avionics computers is the result of the improvements in availability and mission reliability.

1.2 STUDY OBJECTIVES

The principal objectives of this study are as follows:

1. Develop a methodology for conducting tradeoffs of fault tolerance and diagnostic capability of distributed networks versus availability, mission reliability, and life-cycle cost (LCC).
2. Acquire and develop the technical and cost data relevant to the tradeoffs for a state-of-the-art distributed avionics computer network.
3. Validate the methodology by conducting the tradeoff to select an optimum cost-effective design for a representative avionics system.
4. Quantify the sensitivity of availability, mission reliability, and LCC to the levels of fault tolerance and diagnostic capability for a representative platform.
5. Extrapolate the possible consequences of future developments in system architecture, in hardware and software technology, and in avionics subsystems.

1.3 METHODOLOGY

The overall methodology being used in this study is to initially define a number of good candidate fault tolerant computer configurations, and then use established analysis techniques to select the best candidate from the set of good candidates. The steps which implement this methodology are as follows:

1. Define the application requirements for an advanced tactical fighter.
2. Identify the building block options (processors, memories, and buses) from which the candidate configurations can be constructed, and the error handling techniques that might be included.
3. Establish relevant characteristics and data about these building blocks.
4. Establish the necessary tools for measuring the candidate systems.
5. Define the specific objectives against which candidate systems will be evaluated.
6. Define the measures to be used to evaluate the candidate systems.
7. Define one candidate system and analyze it.

8. Evaluate the candidate system against the objectives.
9. Iterate Steps 7 and 8, postulating systems that are better at meeting the objectives.
10. Select the system(s) that best meet(s) the objectives.

This general methodology is diagrammed in Figure 1-1 and is the underlying strategy that coordinates and unifies all of the individual tasks in this program.

The study has been organized into two phases. Phase I develops basic requirements, data, and tools (corresponding to Steps 1-4 in Figure 1-1), while Phase II performs the tradeoff analysis (Steps 5-10). This Interim Report presents the results of the Phase I study.

Section 4 of this report presents the detailed plan for Phase II of the study. However, a generalized description of the Phase II methodology will be given here as an aid to better understanding of the Phase I direction and results.

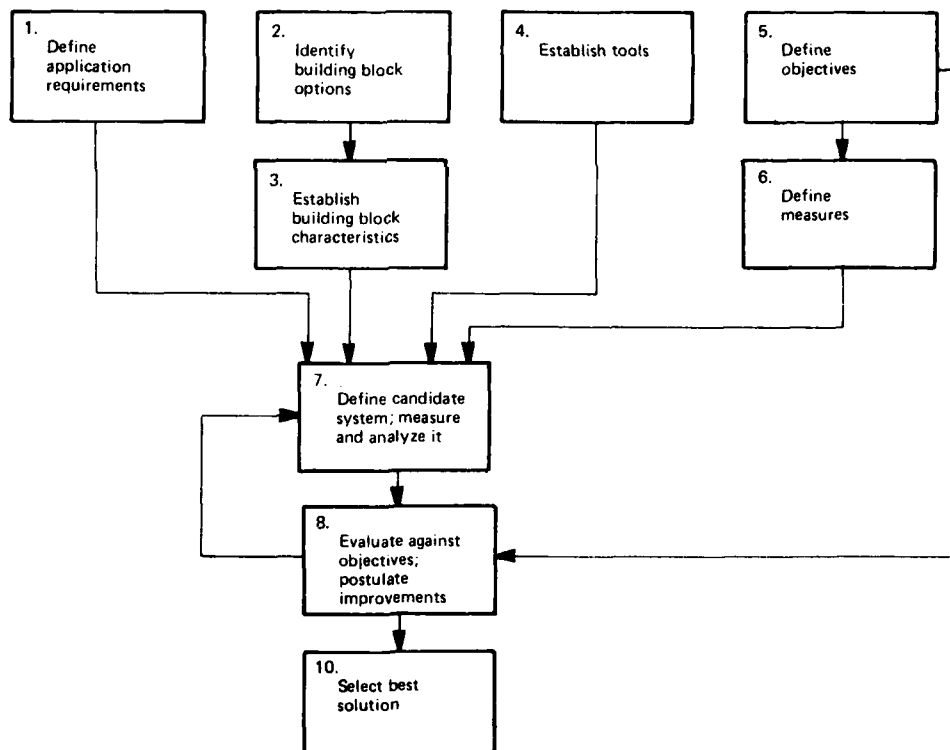


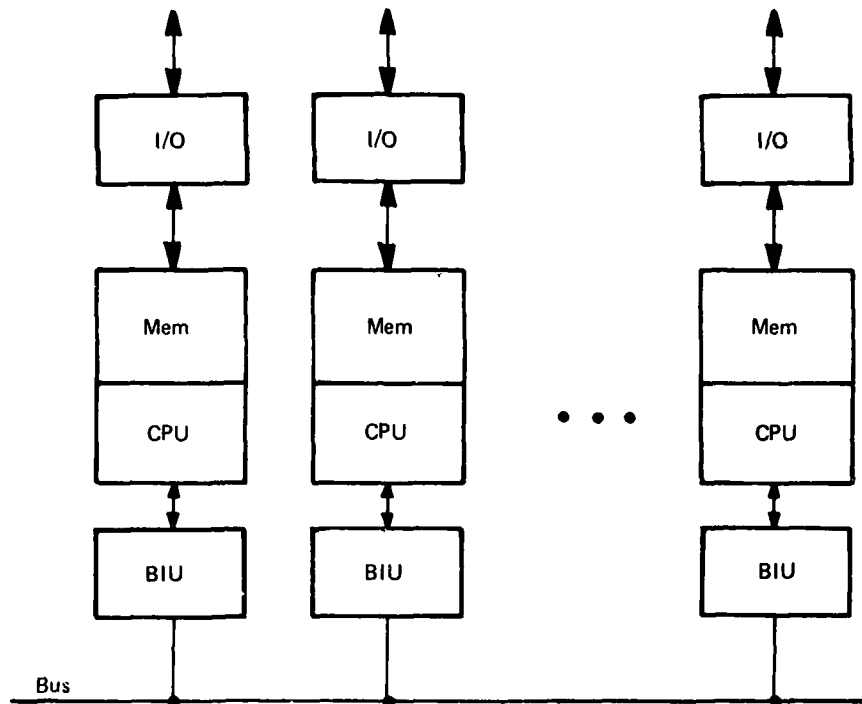
Figure 1-1. General Methodology

The principle objective in Phase II is to establish the cost-effectiveness of various computer fault tolerance techniques. Here "cost" is measured in terms of life cycle cost of the avionics suite (which includes a distributed computer network) for an advanced tactical fighter, and "effectiveness" is measured in terms of the probability of mission success for the avionics suite. The fault tolerance techniques of interest are limited (by the statement of work) to the distributed computer network.

The evaluation is performed in the following sequence of steps:

1. A distributed computer network configuration must be selected. The generic form of such a network is shown in Figure 1-2. Specific processors, memories, and I/O interfaces must be defined, along with a specific system bus and the associated BIUs.
2. The computing load for the advanced tactical fighter application must be partitioned, and the various functions assigned to specific processors.
3. A specific set of error handling techniques must be selected for inclusion in the computer network. These may be error checkers or retry techniques within a particular block of the configuration, or they may be various types of module redundancy.
4. Since the building blocks that make up the configuration are generally known, off-the-shelf elements, basic cost, performance, and reliability data exists for each of them. However, these data must be modified to reflect the addition of these error handling techniques.
5. This final configuration must be analyzed to ensure that the set of computers in fact meets the computational load requirements established for the application, and that the system bus can handle the required traffic without adversely affecting the computing performance. If a specific configuration does not meet these requirements, it must be modified so that it does.
6. The LCC and the probability of mission success of this configuration can then be determined.

This sequence of steps yields one data point. The sequence can be repeated for various selections of error handling techniques, and for various computer network configurations. The resulting data base must then be analyzed to determine which error handling techniques appear to be cost-effective and which do not.



Legend:

CPU	—	Central processing unit
Mem	—	Main memory of the CPU
Bus	—	System bus for the computer network
BIU	—	Bus interface unit
I/O	—	Input/output interface to the avionics subsystems

Figure 1-2. Generic Distributed Computer Network

Section 2
PHASE I TASK RESULTS

In accordance with the original statement of work for this study (Ref. 1), six tasks were established for the Phase I activity. These six tasks were defined to accomplish the first four steps of the methodology shown in Figure 1-1. Table 2-1 summarizes how the tasks relate to the steps in the methodology.

TABLE 2-1. APPLICABILITY OF PHASE I TASKS

Tasks	Appli- cation Req'ts	Building Block Options	Building Block Character- istics	Tools
1.1 Computational Requirements	X			
1.2 Computer Reliability		X	X	
1.3 Error Types		X	X	
1.4 Bus Control Procedures		X		
1.5 Incremental Cost Analysis			X	
1.6 Survey of Modeling Techniques				X

The following listing provides a brief description of each of the six tasks, and shows how the task results relate to the overall study. Subsequent parts of Section 2 present the detailed results from each of these tasks:

1. Task 1.1 - Computational Requirements - The F-15 was selected as the advanced tactical fighter to be used as the baseline application in this study. A description was prepared of the complete avionics suite that currently exists on the F-15. Twelve separate functions were identified and described, and the computational requirements were defined for each (i.e., computer word size, CPU computation rate, and memory size). This separation will be needed in Phase II to partition the functions across the various processors in any new distributed processing configuration. Two such configurations are described below. In addition, the communication requirements are given between each pair of functions, and between each function and its associated avionics subsystem.

These data will be used in the performance modeling to ensure that the processor network defined in a particular configuration supports this F-15 application.

2. Task 1.2 - Computer Reliability - In this task a number of existing, off-the-shelf processors were selected as candidate building blocks to be used in formulating distributed network configurations in Phase II. Quantitative description data is provided about each, particularly including MTBF for each processor. Failure-rate data are also given for other typical subassemblies, such as memories, power supplies, and LSI support modules. These failure-rate data will be used in Phase II for the system reliability analysis for the various configurations that are analyzed. Hardware sizing data will be used to establish life-cycle costs for the different configurations.
3. Task 1.3 - Error Types - For the purposes of this task, each processing element in a distributed computer network is considered to be divided into four major functions; namely, CPU, memory, system bus interface, and I/O interface. For each of these functions, a table is developed of the various types of errors that can occur within that function. Then, for each error type, one or more checkers are identified, which are capable of detecting that error. Estimates are given of the detection effectiveness (i. e. , percentage of errors detected) and coverage (percent of hardware coverage) for each checker, as well as indications of possible retry and recovery actions. The various checkers that are identified are candidates for inclusion in various configurations in Phase II, and for evaluation.
4. Task 1.4 - Bus Control Procedures - The critical hardware element for achieving effective performance and fault tolerance in a distributed computer network is the system bus. Eight buses were analyzed, including the three specifically called for in the statement of work. Twenty criteria were established for evaluating these buses, and the eight candidates were scored accordingly. As a result of this evaluation four of the eight buses have been selected for consideration in Phase II. These will be used as building blocks in the various network configurations. In addition, more detailed design studies and quantitative evaluations will be made, to ultimately recommend an optimum bus configuration at the conclusion of the Phase II study.
5. Task 1.5 - Incremental Cost Analysis - Each of the various error checkers identified in Task 1.3 is analyzed for various cost factors. These cost factors must be applied appropriately to the different configurations to be evaluated in Phase II.
6. Task 1.6 - Survey of Modeling Techniques - Three basic evaluation tools will be needed during the Phase II study. These evaluation tools are as follows:
 - a) Performance Model - This model will be used to verify that a specific computer network configuration will fulfill

the processing requirements established in Task 1.1, and that the system bus will handle the communication traffic resulting from the corresponding partitioning of the application.

- b) Life Cycle Cost (LCC) Model - This model will be used to derive the total avionics suite LCC for each of the configurations evaluated.
- c) Reliability Model - Two levels of reliability modeling will be required in Phase II. One is to determine new failure rates of the building blocks as a result of adding more error-handling facilities. The other is to determine the probability of mission success for the avionics suite with each of the configurations evaluated, including various types of configuration redundancy within the computer network.

This task looks at various approaches to each of these modeling requirements, and selects the one that will be used during Phase II.

2.1 TASK 1.1 - COMPUTATIONAL REQUIREMENTS

This subsection presents baseline functional, computational, and interface requirements for use in defining the Advanced Tactical Avionics System (ATAS) fault tolerant distributed computing network, and evaluating its performance and reliability. These requirements are based on portions of the F-15A avionics system which are currently integrated through a multiple computer and data bus network. The ATAS requirements include a JTIDS Class 2 communication, navigation and identification terminal which is planned for incorporation in the F-15A in the mid-1980 time frame.

Functional requirements, interfaces and data presented in this report are IBM estimates based on experience in the F-15, A-7D/E, and AC-130E programs, and data from available literature.

This subsection contains the following:

1. A brief description of the F-15A avionics system which served as the baseline for the study
2. A description of each of the 12 processing functions defined for ATAS and their interrelationships
3. A summary of processing requirements derived for the 12 ATAS processing functions
4. A description of two candidate ATAS fault tolerant computer network configurations
5. A brief description of the use of functional redundancy in current day fighter aircraft.

The material contained in this section was summarized from Reference 3. That report contains flow charts and more detailed descriptions

of the 12 processing functions, as well as a detailed definition of the processing requirements and intercommunication requirements, which are only summarized here. It also contains operational reliability block diagrams showing specific hardware backup modes as designed into the F-15A avionics system.

2.1.1 F-15A AVIONIC SYSTEM DESCRIPTION

The F-15A avionic system augmented with a JTIDS Class 2 terminal was selected as the baseline for the ATAS. This selection was made because the avionic system architecture and partitioning of the multi-computer network is representative of all current-day fighters. Further, a detailed relevant data base of F-15 processing and information transfer requirements was developed by IBM for the McDonnell Douglas Aircraft Company. This data base was used to develop central computer baseline computational requirements.

The major equipment comprising the F-15A avionics system (Figure 2-1) are the radar, armament control set, lead computing gyro, head-up display, indicator group, air data computer, central computer, UHF receiver/transmitter, inertial navigation set, radar warning receiver, internal countermeasures set, and TACAN. Also included in the system are the flight instruments, IFF equipment, built-in test panels, and an aircraft structural integrity recorder. Avionic subsystems are interconnected by a dual-redundant, digital multiplex data bus system, and dedicated lines for video, electrical power, and functional backup.

A JTIDS Class 2 terminal is included in this baseline definition of the ATAS to reflect a key modification being planned for the F-15A system. Other electronic equipment such as the automatic flight control set and air inlet controller are essentially isolated from the above equipment in the F-15 due to their flight safety criticality and are, therefore, not part of the avionic system as defined in this report.

The baseline F-15 multiple computer network is illustrated in Figure 2-2. General purpose computers are imbedded in the JTIDS Class 2 terminal, fire control radar, radar warning receiver, inertial navigation system, air data system, display signal data processor, and armament control set. The inertial navigation set currently incorporates a special purpose digital differential analyzer (DDA). For purposes of this study, inertial navigation system computational requirements were developed for a general purpose processor implementation of the equivalent DDA functions.

Communications between the eight general purpose computers in the network are accommodated through a McDonnell Douglas Aircraft H009 data bus. The data bus has four serial channels. Two channels are used in normal operations. The two remaining channels provide a redundant backup to improve the survival probability in case the aircraft is hit by enemy weapons. Data transfers are at a 1 MHz rate.

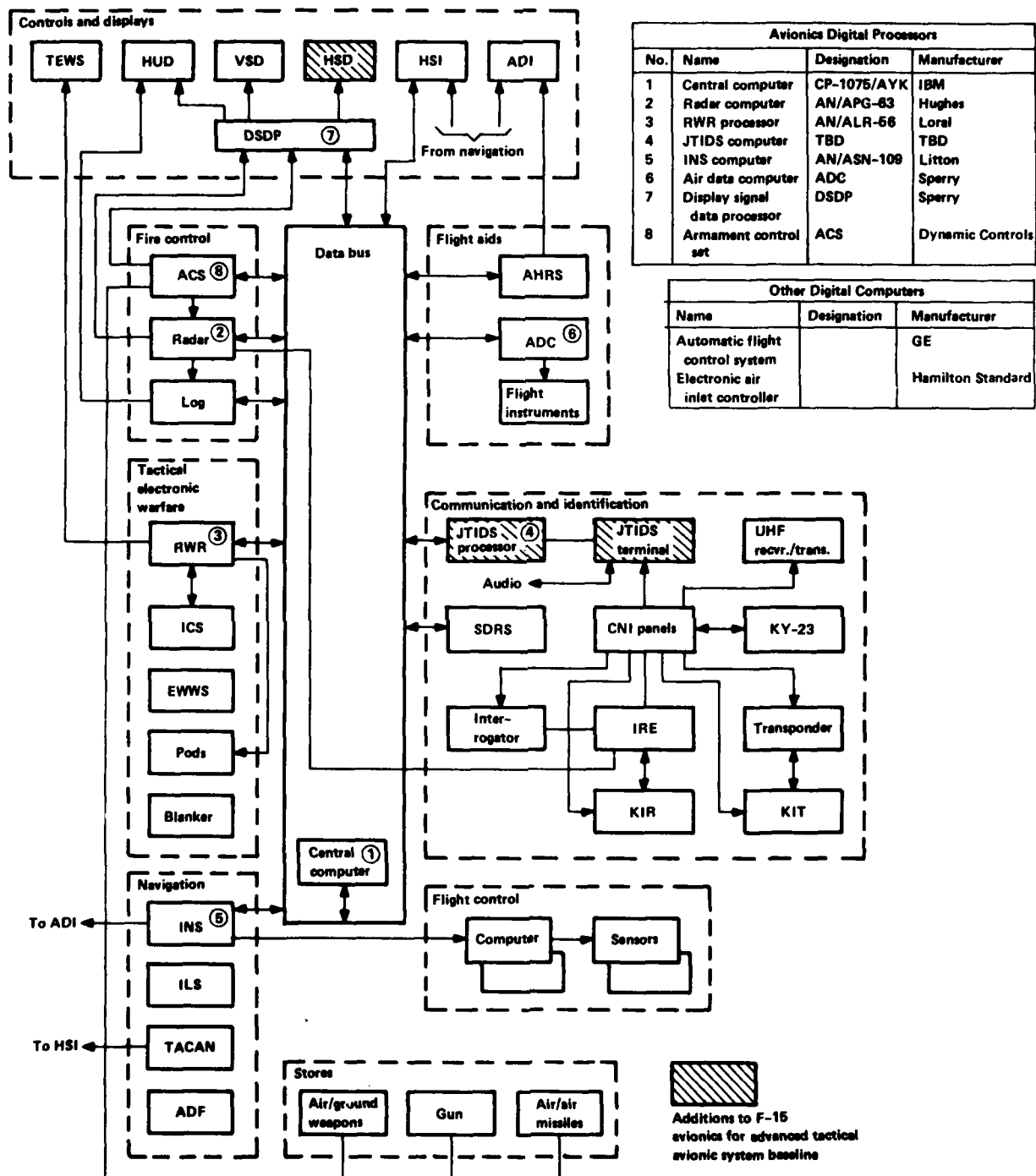


Figure 2-1. Representative F-15 Avionics Functional Block Diagram

Additional computers exist in the automatic flight control system and air inlet controller. These subsystems are not part of the avionic system described here and are not interfaced on the avionics multiplex data bus.

Special purpose processors are imbedded in the JTIDS Class 2 terminal, fire control radar, and radar warning receiver. These processors perform high-speed signal data preprocessing before further processing by the general-purpose computers in each of these subsystems. The special purpose processors are not included in the distributed computer network because of their high processing speeds and special hardware characteristics.

2.1.2 ATAS FUNCTIONAL ELEMENTS

The purpose of the ATAS is to enable delivery of air-to-air and air-to-ground weapons on targets that must be detected, identified, acquired, tracked, and destroyed by the pilot using sophisticated sensors and weapons.

To satisfy the primary requirement to provide accurate air-to-air and air-to-ground weapon delivery during all weather conditions under one-man operation, numerous and varied computational tasks are required.

The ATAS computational tasks fall into three general categories:

1. Sensor-oriented computations
2. Mission-oriented computations
3. Computer network management computations.

Sensor-oriented computations are those independent computations, such as sensor coordinate transformations, platform management, and signal processing, which are peculiar to a particular sensor or display and not dependent on information from other sensors. Mission-oriented computations are directly related to performing the mission and are dependent on the integration of information from several avionics subsystems. Computer network computations are those computations required to control tasks within each network computer, control input/output operations, perform built-in-tests, reconfigure network functions according to the mode selected by the pilot, and perform automatic reconfiguration when malfunctions are detected.

The total computational requirements have been partitioned into 12 major functions for the ATAS. Each major function includes executive, input/output, and built-in test functions required to manage the computer network.

Figure 2-3 illustrates the key relationships between the 12 ATAS data processing functions and their associated sensors, displays, controls, and armament devices. These relationships are summarized in the following paragraphs.

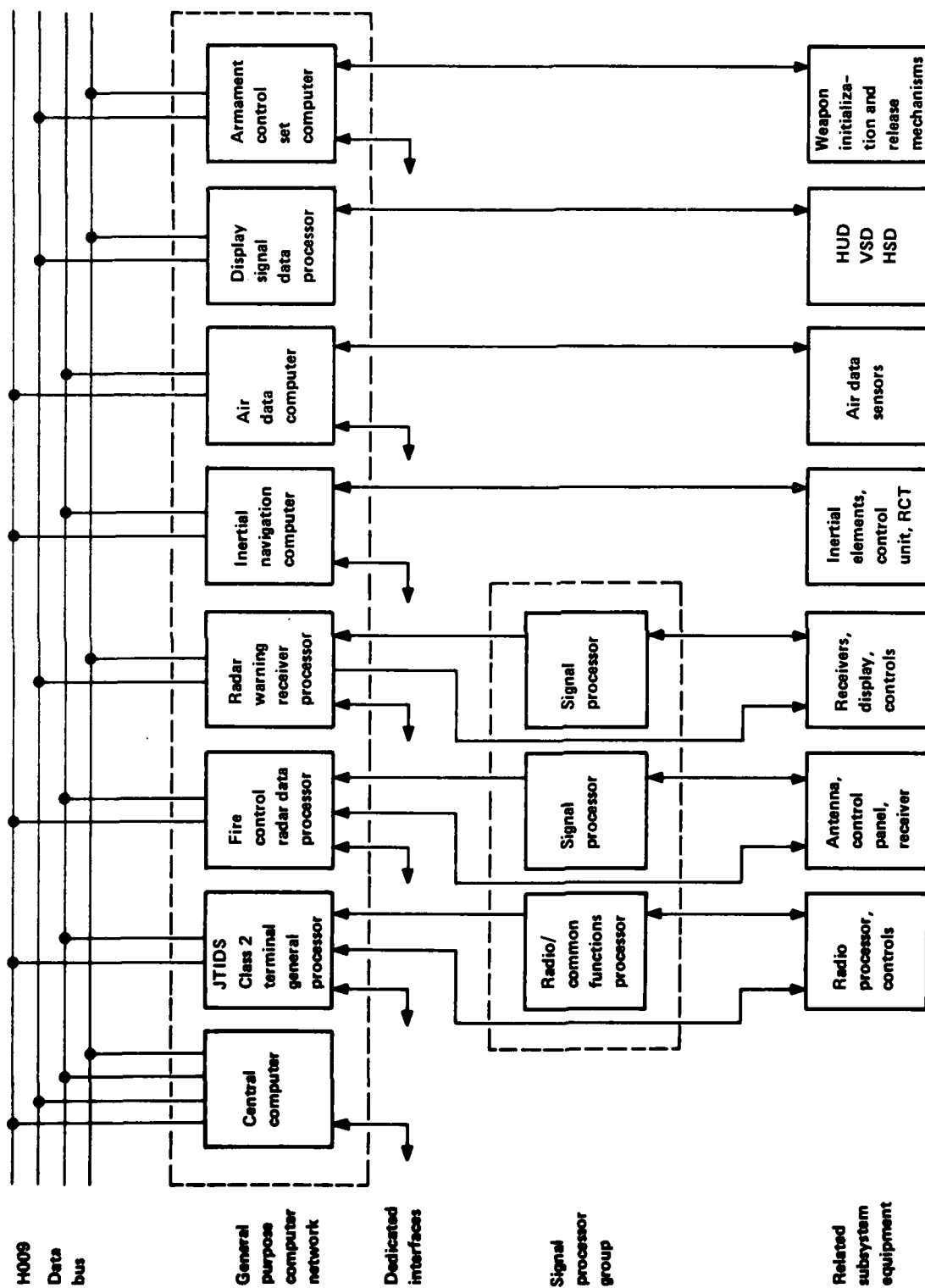
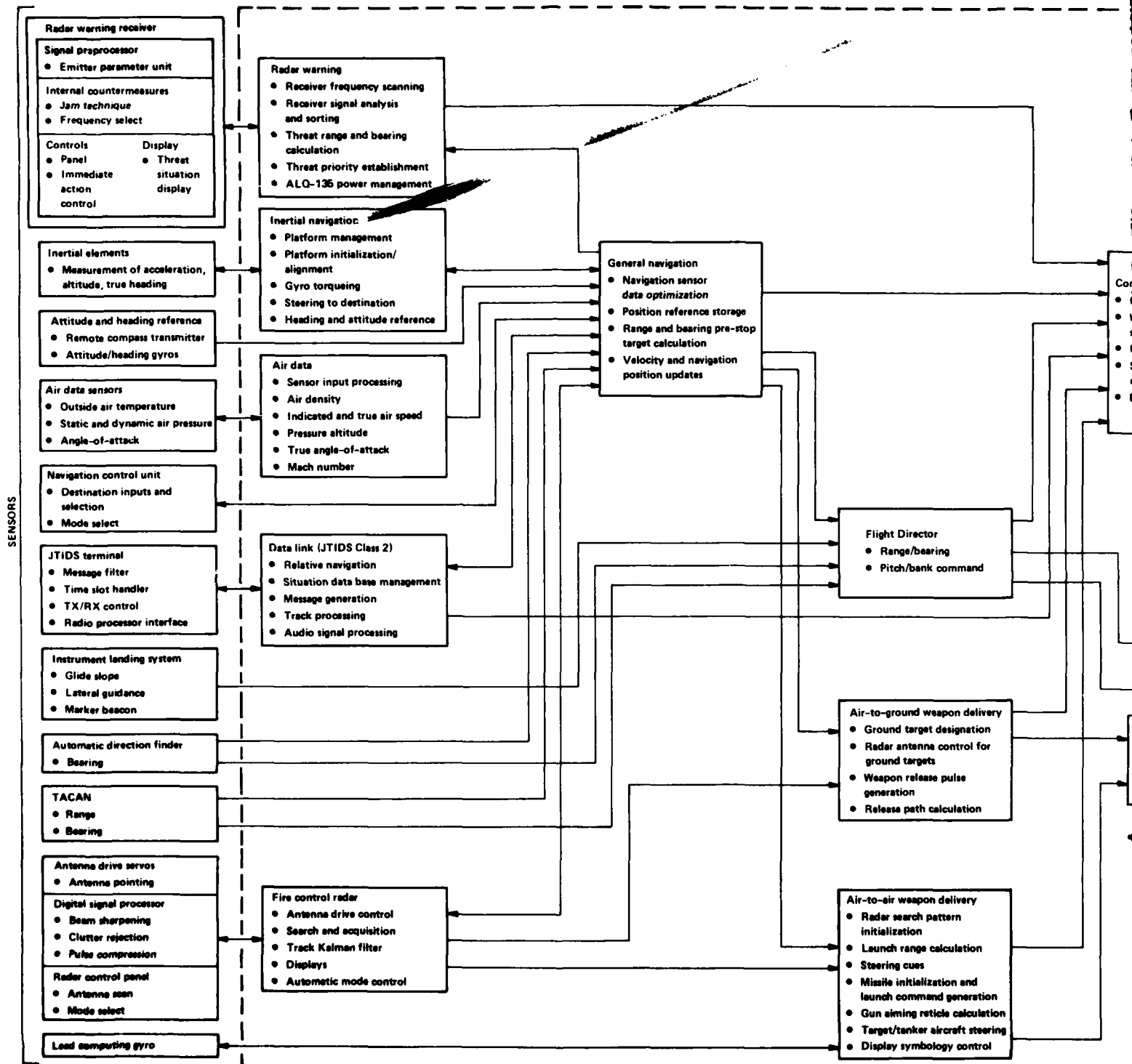


Figure 2-2. F-15 Baseline Multicomputer Network



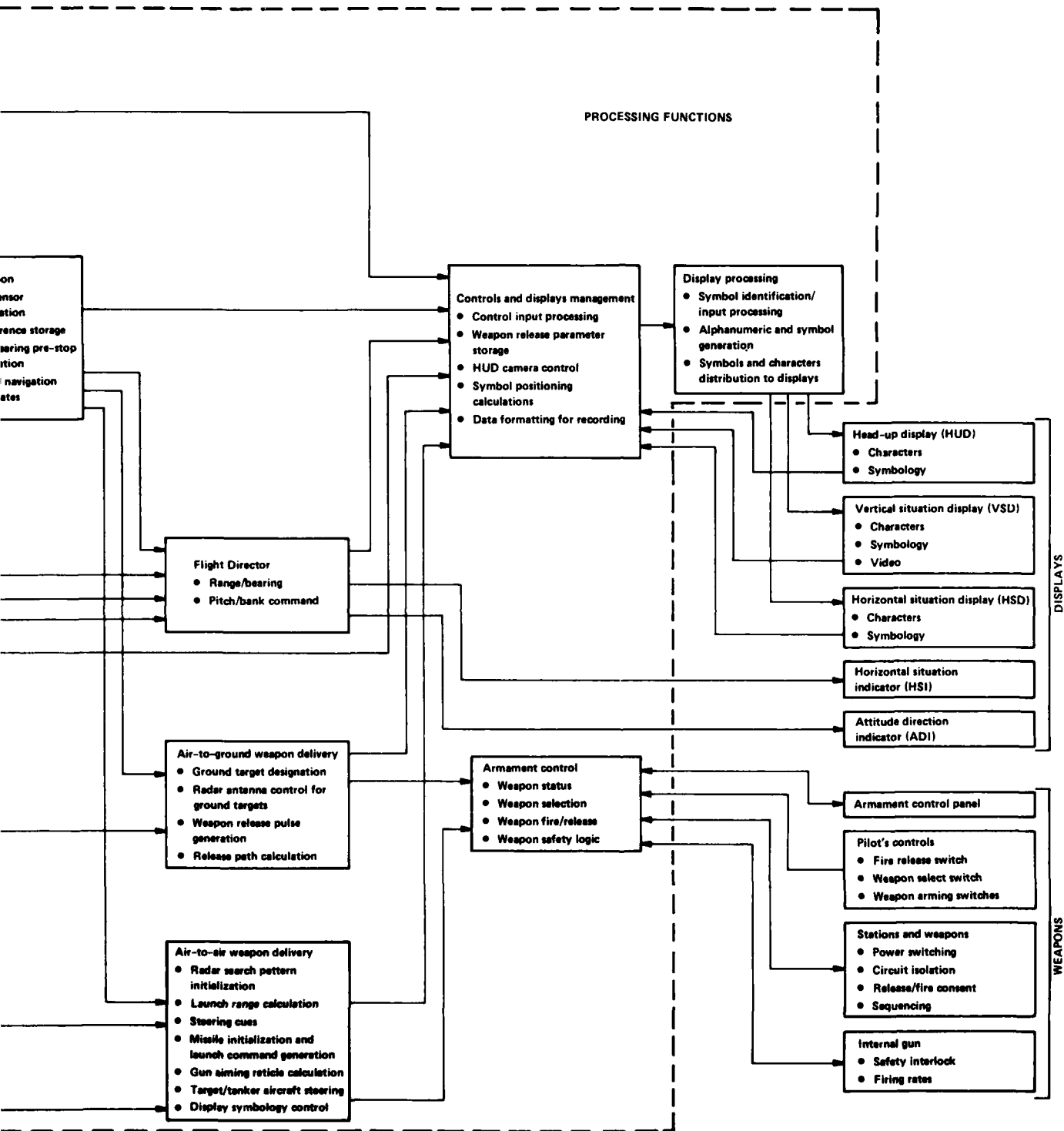


Figure 2-3. ATAS Data Processing Function Interfaces

2.1.2.1 Inertial Navigation

The aircraft initial present position, stored destinations, and pilot-controlled position updates are provided by the general navigation function to the inertial navigation function. Between updates the inertial navigation function continuously provides aircraft present position, velocity, attitude, and destination steering information to the general navigation function.

2.1.2.2 Air Data

The air data function processes air pressures, temperature, and angle-of-attack measurements, and provides calibrated airspeed, altitude, and angle-of-attack data to the general navigation function and flight instruments. The general navigation function provides sea-level barometric pressure to the air data function to initialize altitude calculations.

2.1.2.3 General Navigation

The general navigation function receives navigation data from the following sources and selects the best available data for aircraft navigation use:

1. Inertial navigation function provides position, velocity, attitude and steering data
2. Air data function provides airspeed, altitude, and angle-of-attack data
3. Attitude and heading reference system provides backup attitude and heading data
4. JTIDS Class 2 terminal provides accurate position updates relative to other members of the JTIDS network
5. Fire control radar provides doppler velocity and position updates when initiated by the pilot
6. TACAN provides range and bearing to cooperative ground-based TACAN stations for use in aircraft position update or steering to the TACAN station
7. Automatic direction finder provides bearing information to any radio transmission received by the automatic direction finder
8. Navigation control unit provides means for the pilot to enter aircraft initial position and destination coordinates and sea-level pressure as well as heading, winds, altitude, airspeed, etc. for backup navigation purposes.

The general navigation function provides the best available navigation data or other data selected by the pilot to the following functions:

1. Controls and Displays Management - All navigation and steering parameters of interest to the pilot, as well as the states of all elements used in the navigation function.
2. Air-To-Air and Air-To-Ground Weapon Delivery - position, velocity, and air data required to determine the necessary conditions for weapon release. Coordinates of preassigned ground targets are also provided for automatic weapon delivery against these targets.
3. Flight Director - Range and bearing data are provided to assist the pilot to fly to or from a selected destination.
4. Radar Warning Receiver - Aircraft position and altitude data for use in determining threat coordinates.

2.1.2.4 Flight Director

The flight director function receives range and bearing from the general navigation function and TACAN, bearing from the automatic direction finder, and lateral steering and glide slope errors from the instrument landing system.

The flight director provides range, bearing, command heading and steering error for the pilot selected destination, target or TACAN station for display by the horizontal situation display and indicator. The flight director also provides bank steering to these selected destinations, as well as pitch and bank steering commands for instrument landing approaches to the head-up display, vertical situation display, or attitude direction indicator.

2.1.2.5 Fire Control Radar

The fire control radar general purpose processor function receives airborne target relative position and velocity from the radar's digital signal processor and provides this data to the air-to-air weapon delivery functions. The radar processor also provides target range, azimuth, and elevation to the air-to-ground weapon delivery function. The radar processor also provides range, azimuth, elevation, and relative velocity with respect to ground stations to the general navigation function for use in position and velocity updates. The radar processor function receives initial antenna pointing commands from the air-to-air and air-to-ground weapon delivery functions, and velocity data from the general navigation function. The fire control radar control panel, throttle-mounted and stick-mounted radar control switches provide inputs to the fire control radar data processor to select the radar's operating mode.

2.1 2.6 Air-To-Air Weapon Delivery

The air-to-air weapon delivery function initializes the fire control radar search pattern and receives relative target position and velocity from the fire control radar function. The air-to-air weapon delivery function provides weapon launch cues and status to the controls and displays management function for display on the head-up and vertical situation displays. The air-to-air weapon delivery function provides initialization commands for the selected air-to-air missile via the armament control set and receives the status of these missiles from the armament control set.

2.1.2.7 Air-To-Ground Weapon Delivery

The air-to-ground weapon delivery function controls the pointing of the fire control radar antenna. This function provides steering cues and weapon status to the controls and displays management function according to the selected weapon delivery mode, and provides weapon release pulses when weapons are released automatically. The air-to-ground weapon delivery function receives designated target position data from either the fire-control radar or the HUD and aircraft position and velocity information from the general navigation function for use in weapon delivery calculations.

2.1.2.8 Armament Control

The armament control function receives weapon selection, initialization and release information from the air-to-air and air-to-ground weapon delivery functions. It then provides fuzing and arming commands, release sequencing and emergency jettison. The armament control function receives status information from the weapons and their respective weapon racks and provides this status to the weapon delivery functions.

2.1.2.9 Radar Warning

The radar warning function receives aircraft position and attitude information from the general navigation function for use in determining received threat radar signal locations. The radar warning function controls a warning tone to the intercommunications set when the function determines that the aircraft is in imminent danger.

2.1.2.10 JTIDS

The JTIDS general processor provides relative navigation data (based on derived ranges to other JTIDS network members) to the general navigation function for position update, and receives the aircraft's

position and status data from the general navigation function for dissemination to other members of the JTIDS network. The JTIDS function provides uplink commands and data to the controls and displays management function for display on the horizontal situation display. The JTIDS function also accommodates two-way analog voice interfaces with the aircraft's intercommunication set.

2.1.2.11 Controls and Displays Management

The controls and displays management function receives data and status information from the general navigation, radar warning, air-to-air weapon delivery, air-to-ground weapon delivery, JTIDS processor, flight director, and various cockpit controls. The controls and displays management function formats the characters and symbology resulting from the data and status information.

The function provides character and symbology location and orientation instructions to the display signal data processor for display on the head-up, vertical situation or horizontal situation displays according to the operating mode selected by the pilot. The controls and displays management function provides stored weapon ballistics parameters to the air-to-air and air-to-ground weapon delivery functions and provides formatted aircraft status and weapon release parameters to the signal data history recorder. The controls and displays management function also controls the head-up display camera when firing weapons.

2.1.2.12 Display Signal Data Processing

The display signal processor function receives character and symbology position and orientation data from the controls and displays management function, generates the corresponding characters and symbology, and distributes the resulting analog stroke writing signals to the head-up, horizontal situation, and vertical situation displays. The display signal data processor also accepts backup data from the lead computing gyro and data from the flight director for display on the HUD.

2.1.3 PROCESSING REQUIREMENTS SUMMARY

Estimated computational speed and storage requirements for each of the 12 ATAS processing functions are summarized in Table 2-2. The speed (KOPS) and storage (K words) data were derived from IBM's estimates of processing requirements for the F-15A avionics system functions and from an IBM analysis of the JTIDS processing requirements. These data include the basic processing functions, executive, built-in-test, input/output, and reserves for each processing function. The reserves include 30% for storage words and 60% for speed requirements. Storage and speed requirements are made under the assumption that

TABLE 2-2. ATAS COMPUTATIONAL REQUIREMENTS SUMMARY

Processing Function	Word size (bits)			Intercommunications (words per second)												Input/output (words per second)	
	KOPS	Storage (K words)	Inertial navigation	Air data	General navigation	Flight director	Fire control radar	Air-to-air weapon	Air-to-ground weapon	Armament control	Radar warning	JTIDS terminal	Controls and displays	Display processing	Totals		
Inertial navigation	16	130	12			260									260	260	
Air data	16	19	2			220									220	125	
General navigation	16	20	6	40	20		15	75		360		35	85	95		725	195
Flight director	16	29	3										190			190	145
Fire control radar	24	743	41			25		300								325	340
Air-to-air weapon	16	168	11					100		280			680			1,060	140
Air-to-ground weapon	16	213	9					43		20			520			583	-
Armament control	16	21	3					200	131							331	665
Radar warning	16	266	40									1				1	11,640
JTIDS terminal	16	1305	92			3 1/2							225			225 + 3/12	-
Controls and displays	16	66	4									84		1795		1,879	25
Display processing	32	281	10										25			25	Analog
Total															5,824		

each function is executed totally within its own processor. Component requirements will be reduced slightly when two or more functions are executed in the same processor, as illustrated in subparagraph 2.1.4.

Speed and storage requirements were adjusted to account for word size differences between the processor used in the F-15A avionics system and the 16-bit word length processors assumed for ATAS.

Table 2-2 also summarizes intercommunication rates between the various ATAS processing functions. These intercommunications should be readily accommodated by a standard data bus such as the MIL-STD-1553A. The input/output column gives additional communication data rates that, in the F-15A configuration, are accommodated by dedicated interfaces. One example of this is the communications between the inertial navigation sensors and the inertial navigation computer, where special purpose dedicated interfaces are employed.

2.1.4 CANDIDATE ATAS FAULT TOLERANT COMPUTER NETWORK CONFIGURATIONS

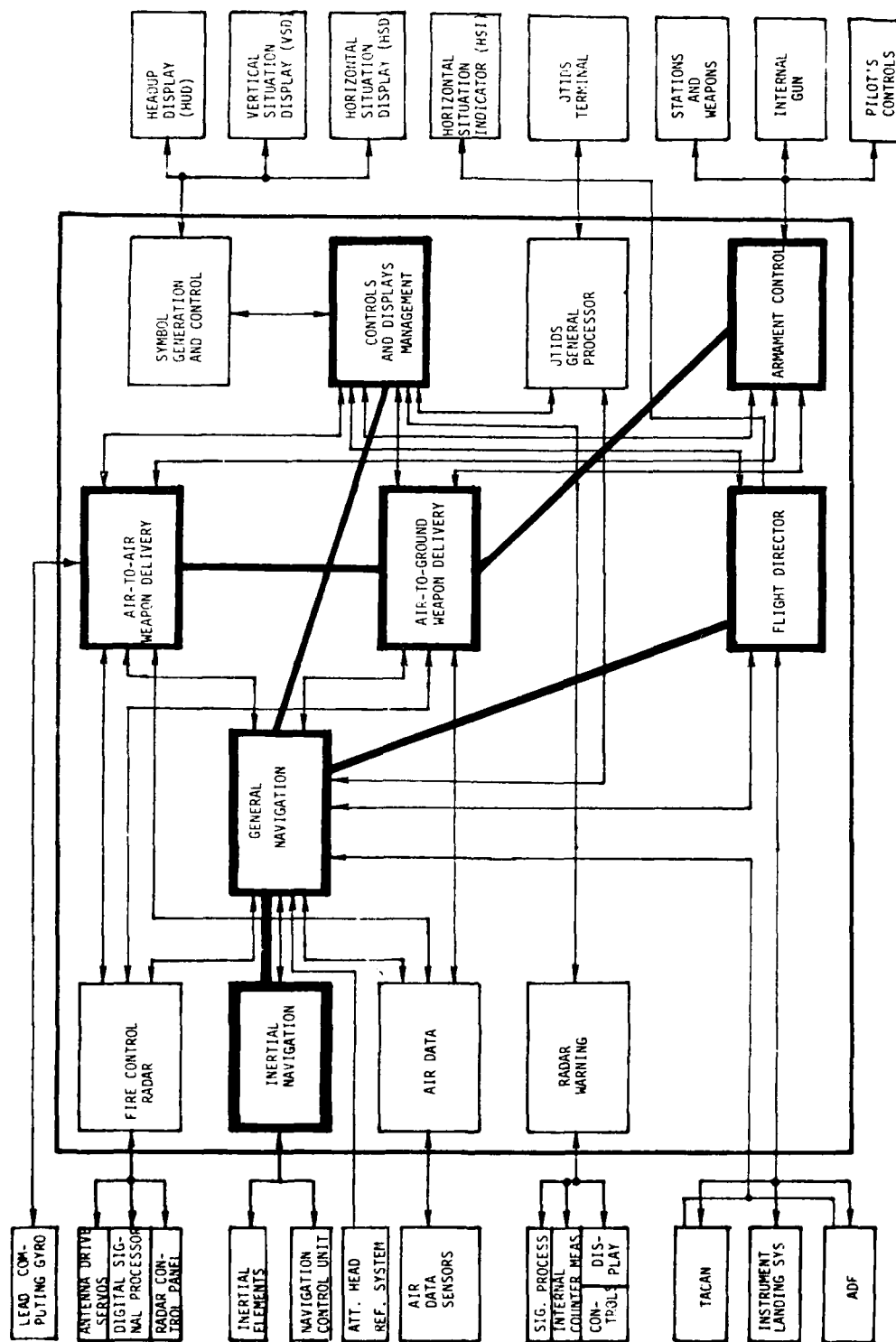
This paragraph presents two candidate distributed computer network configurations for the ATAS application. The two candidates are partitioned such that each computer in the network may perform a complete ATAS function or a complete group of ATAS functions.

The processing requirements for these two candidates include estimates of executive, input/output and built-in test based on the existing (F-15) architecture. These estimates may need to be increased due to incorporation of a distributed executive and fault tolerance/network reconfiguration features.

2.1.4.1 Distributed Central Computer Network (Candidate Number 1)

For the distributed central computer network candidate, the traditional central computer is deleted and its functions are combined with the inertial navigation and armament control functions. As illustrated in Figure 2-4, the general navigation, inertial navigation, flight director, and controls and displays management functions are consolidated into a "navigation" computer. Similarly, the air-to-air and air-to-ground weapon delivery function and armament control function are consolidated into a weapon delivery/armament control computer.

Requirements for the seven computers in this candidate network are summarized in Table 2-3. In this table, the navigation and weapon delivery/armament control computers are assumed to have a 16-bit word length.



Note: Weapon Delivery/Armament Control includes Air-To-Air and Air-To-Ground
 Weapon Delivery Functions and the Armament Control Set Function
 Navigation Computer includes General Navigation, Flight Director and Controls
 and Displays Management Functions.

Figure 2-4. Distributed Central Computer Network (Candidate Number 1)

TABLE 2-3. DISTRIBUTED COMPUTER NETWORK CONFIGURATION 1 REQUIREMENTS

Navigation (16 BITS)		
Function	KIPS	K
Inertial Navigation	130.4	12.2
General Navigation	20.4	5.4
Flight Director	28.9	2.4
Control/Display, Management	65.9	3.7
	245.6	23.7

Weapon Delivery Armament Control (16 BITS)		
Function	KIPS	K
Armament Control	21.1	2.7
Air-to-Ground	212.5	8.5
Air-to-Air*	167.6	10.9
*Not in worst Case Path	233.6	22.1

Air Data (16 BITS)		
Function	KIPS	K
Air Data	19.2	2.3

Display Processor (32 BITS)		
Function	KIPS	K
Display Generation	280.8	9.6

JTIDS Terminal (16 BITS)		
Function	KIPS	K
Adaptable And Tailored Functions Processing	1,305	119.9

Radar (24 BITS)		
Function	KIPS	K
Radar Processor	743	40.7

Radar Warning (16 BITS)		
Function	KIPS	K
Radar Warning Processor	265.9	40.3

KIPS - Thousands of instructions per second
 K - Thousands of memory words
 Data include 30% storage and 60% speed reserves
 Data are adjusted for word size

2.1.4.2 Widely Distributed Computer Network (Candidate Number 2)

For the widely distributed computer network candidate, each of the 12 ATAS computer network processing functions is contained in a separate computer, as illustrated in Figure 2-5.

Processing requirements for this candidate are summarized in Table 2-4. This configuration utilizes ten 16-bit computers, one 24-bit computer (fire control radar) and one 32-bit computer. Nine of the ten 16-bit computers could be implemented with standardized, low-cost computers using today's technology. The JTIDS terminal computer storage and speed requirements are significantly greater than the capabilities of single current low-cost computers.

2.1.5 FUNCTIONAL REDUNDANCY

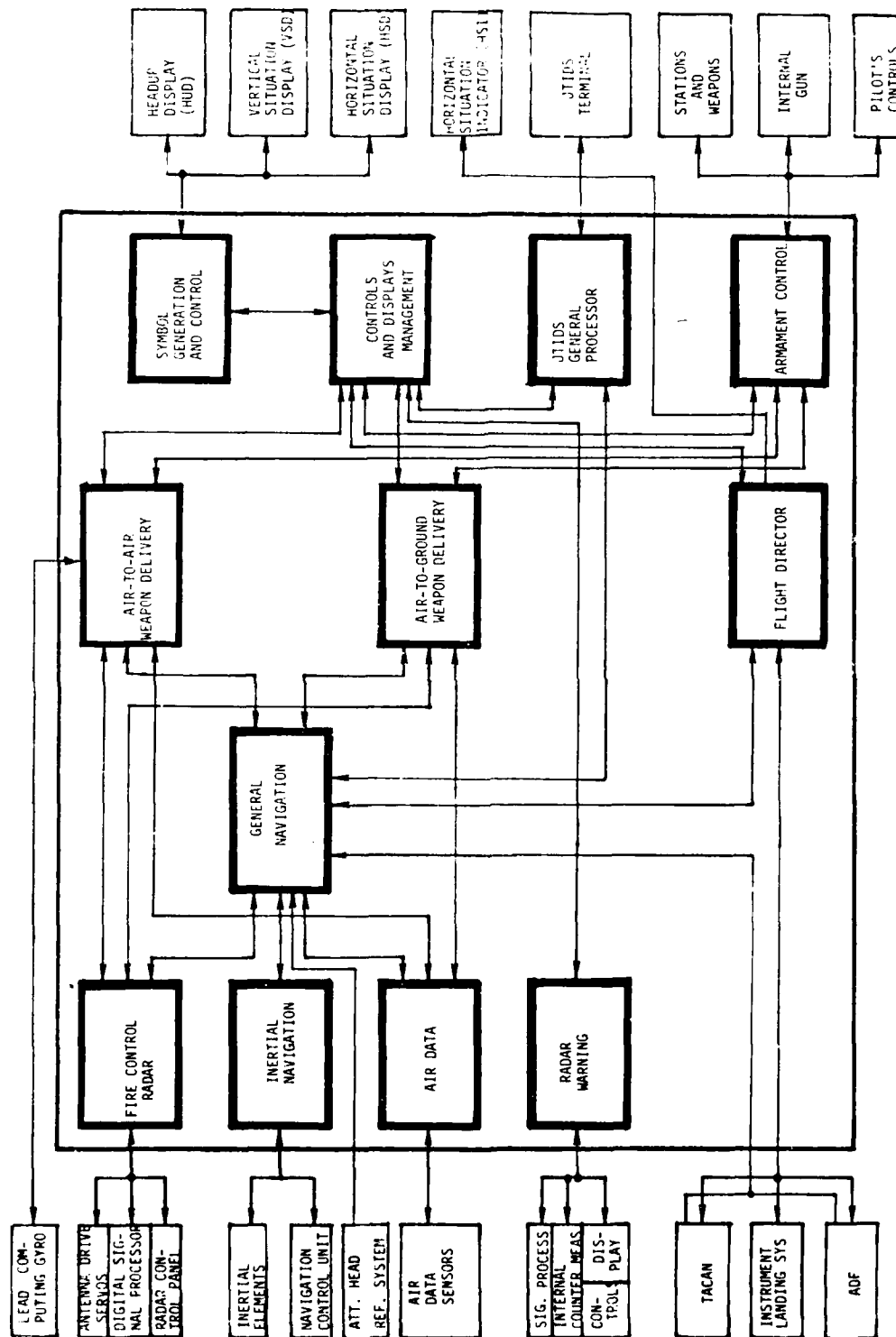
Functional redundancy is used extensively in all modern tactical fighters such as the F-15 to provide maximum safety for the pilot and aircraft, and to improve the probability of mission success. These major operational requirements are satisfied in the avionic system by providing a resistance to equipment failure (fault tolerance) and resistance to equipment failure caused by combat (damage tolerance). The complete redundancy design must, therefore, address both fault and damage tolerance criteria, since either can become the dominant requirement depending upon aircraft use and threat environment.

Functional redundancy is achieved in the avionic system by two basic methods:

1. Element Redundancy - Maintains full system capacity and accuracy following loss of a primary element by duplicating critical hardware and software.
2. Back-Up Modes - Results in degraded system performance following loss of a primary element by using remaining hardware and software in an alternative mode, or by using a lower performance hardware/software alternative.

The total redundancy design is usually a composite of redundancy and back-up mode methods. The specific techniques are selected by the system architect during trade off analysis to increase flight safety and mission success capabilities within major implementation constraints, such as avionics weight, power, cooling, and life-cycle cost. In the past, single-seat fighter avionics typically have implemented back-up modes rather than hardware redundancy techniques because of the high installation and cost penalties, and weight and power penalties, that result from duplicate hardware approaches.

Two missions that reflect the operational capabilities of the ATAS are the interdiction and escort missions.



NOTE: Each processing function is performed in a separate computer

Figure 2-5. Widely Distributed Computer Network (Candidate Number 2)

TABLE 2-4. DISTRIBUTED COMPUTER NETWORK CONFIGURATION 2 REQUIREMENTS

<table><tr><td>Inertial Navigation</td><td>16 Bits</td></tr><tr><td>130.4 KIPS 12.2 K</td><td></td></tr></table>	Inertial Navigation	16 Bits	130.4 KIPS 12.2 K		<table><tr><td>Air Data</td><td>16 Bits</td></tr><tr><td>19.2 KIPS 2.3 K</td><td></td></tr></table>	Air Data	16 Bits	19.2 KIPS 2.3 K		<table><tr><td>General Navigation</td><td>16 Bits</td></tr><tr><td>20.4 KIPS 5.8 K</td><td></td></tr></table>	General Navigation	16 Bits	20.4 KIPS 5.8 K		<table><tr><td>Flight Director</td><td>16 Bits</td></tr><tr><td>28.9 KIPS 2.8 K</td><td></td></tr></table>	Flight Director	16 Bits	28.9 KIPS 2.8 K	
Inertial Navigation	16 Bits																		
130.4 KIPS 12.2 K																			
Air Data	16 Bits																		
19.2 KIPS 2.3 K																			
General Navigation	16 Bits																		
20.4 KIPS 5.8 K																			
Flight Director	16 Bits																		
28.9 KIPS 2.8 K																			
<table><tr><td>Fire Control Radar</td><td>24 Bits</td></tr><tr><td>743 KIPS 40.7 K</td><td></td></tr></table>	Fire Control Radar	24 Bits	743 KIPS 40.7 K		<table><tr><td>Air-To-Air Weapon Delivery</td><td>16 Bits</td></tr><tr><td>167.6 KIPS 11.3 K</td><td></td></tr></table>	Air-To-Air Weapon Delivery	16 Bits	167.6 KIPS 11.3 K		<table><tr><td>Air-To-Ground Weapon Delivery</td><td>16 Bits</td></tr><tr><td>212.5 KIPS 8.9 K</td><td></td></tr></table>	Air-To-Ground Weapon Delivery	16 Bits	212.5 KIPS 8.9 K		<table><tr><td>Armament Control</td><td>16 Bits</td></tr><tr><td>21.2 KIPS 2.7 K</td><td></td></tr></table>	Armament Control	16 Bits	21.2 KIPS 2.7 K	
Fire Control Radar	24 Bits																		
743 KIPS 40.7 K																			
Air-To-Air Weapon Delivery	16 Bits																		
167.6 KIPS 11.3 K																			
Air-To-Ground Weapon Delivery	16 Bits																		
212.5 KIPS 8.9 K																			
Armament Control	16 Bits																		
21.2 KIPS 2.7 K																			
<table><tr><td>Radar Warning</td><td>16 Bits</td></tr><tr><td>265.9 KIPS 40.3 K</td><td></td></tr></table>	Radar Warning	16 Bits	265.9 KIPS 40.3 K		<table><tr><td>JTIDS Terminal</td><td>16 Bits</td></tr><tr><td>1,304.7 KIPS 92.3 K</td><td></td></tr></table>	JTIDS Terminal	16 Bits	1,304.7 KIPS 92.3 K		<table><tr><td>Controls And Displays Management</td><td>16 Bits</td></tr><tr><td>65.9 KIPS 1.9 K</td><td></td></tr></table>	Controls And Displays Management	16 Bits	65.9 KIPS 1.9 K		<table><tr><td>Display Signal Data Processor</td><td>32 Bits</td></tr><tr><td>280.8 KIPS 9.6 K</td><td></td></tr></table>	Display Signal Data Processor	32 Bits	280.8 KIPS 9.6 K	
Radar Warning	16 Bits																		
265.9 KIPS 40.3 K																			
JTIDS Terminal	16 Bits																		
1,304.7 KIPS 92.3 K																			
Controls And Displays Management	16 Bits																		
65.9 KIPS 1.9 K																			
Display Signal Data Processor	32 Bits																		
280.8 KIPS 9.6 K																			

KIPS - Thousands of instructions per second
K - Thousands of memory words
Data include 30% storage and 60% speed reserves
Data are adjusted for word size

During an interdiction mission, navigation, tactical electronic warfare, and communication and identification functions are required continuously. TACAN and ADF navigation modes are only available when the fighter is on the friendly side of the forward edge of the battle area (FEBA). Acquisition and target designation are performed in preparation for weapon delivery.

The escort mission similarly requires the fighter to perform navigation, tactical electronic warfare, and communications and identification functions when in flight with the same restrictions on the navigation modes of TACAN and ADF as before. Air-to-air search and target acquisition will be required in all in-flight phases. When air-to-air encounter with enemy fighters is performed, target acquisition, target tracking, and missile and gun engagement are required.

Mission functions performed during interdiction and escort missions include the following:

1. Navigation and guidance
2. Fixtaking
3. Communication and identification
4. Tactical electronic warfare
5. Air-to-ground weapon delivery
6. Air-to-air weapon delivery.

Redundant elements employed in performance of these functions are described below.

2.1.5.1 Navigation and Guidance

The primary navigation sensor is the inertial navigation sensor because of its superior accuracy. Back-up navigation sensors include various combinations of the attitude and heading reference system, air data system, JTIDS terminal (when used for relative navigation or TACAN) and an automatic direction finder. The head-up display is the primary display. Back-up displays include the vertical situation display, horizontal situation display, horizontal situation indicator, and attitude direction indicator.

2.1.5.2 Fixtaking

Sensors used for fixtaking include the radar (in ground map mode), head-up display (for visual observations), and JTIDS (using relative navigation or TACAN). Accuracies of each sensor depend upon observation geometry and knowledge of reference point locations. The radar is the primary fix-taking sensor because of its accuracy, lack of dependency on ground beacons or communication terminals, and its ability to be used in adverse weather conditions.

2.1.5.3 Communications and Identification

JTIDS and the UHF transceiver are the primary communications equipments. The IFF interrogator/transponder is the primary identification device. The UHF transceiver has a back-up UHF receiver. The IFF system does not have a back-up and is required to be operating at all times during flight.

2.1.5.4 Tactical Electronic Warfare

The primary mode equipments currently in use are the radar warning receiver and the internal countermeasures set. The JTIDS terminal will also be used in the primary mode because of its ability to automatically disseminate locations and identifications of all detected threats to all members of the JTIDS network. These threats may be detected by radar warning receivers on any RWR and JTIDS equipped aircraft, or by other means. If either the RWR or the JTIDS terminal fails, the remaining equipment can still provide backup threat radar indications to the pilot.

2.1.5.5 Air-to-Ground Target Designation/Weapon Delivery

The HUD radar (A-G ranging mode) and inertial navigation set are the primary air-to-ground target designation and weapon delivery sensors for the day VFR interdiction mission. The radar (ground map mode) is also the primary sensor at night and in adverse weather if the target presents a recognizable radar return or if the target's coordinates are known with respect to an offset point with a recognizable return. The JTIDS terminal may be used to designate a target if another member of the JTIDS network can accurately provide the target's coordinates in the JTIDS navigation grid.

In the absence of other targeting sensors, the HUD manually depressed reticle may be used, and the weapon released under predetermined speed, altitude, and attitude conditions.

2.1.5.6 Air-to-Air Target Acquisition/Tracking/Weapon Delivery

The radar is the primary air-to-air target acquisition, tracking and weapon delivery sensor. The inertial navigation sensor provides the primary means of stabilizing the radar antenna pointing during aircraft maneuvers. The air-to-air weapon delivery function provides primary mode steering cues and weapon status data through the HUD and VSD for the pilot's use in selecting and releasing the weapon. The air-to-air weapon delivery function is essential for initializing the seeker on the selected air-to-air missile.

The lead computing gyro in conjunction with the HUD allows the pilot to fire the gun if other weapon delivery modes are not available.

2.2 TASK 1.2 - COMPUTER RELIABILITY

This subsection summarizes operating characteristics and operational environment failure rates for a number of existing technology processors.

1. AP-1, AP-101, AP-101C, and AYK-14 are all basically ATR-sized avionics computers, and each represents a packaged processor with memory, I/O and self-contained power.
2. BLP is a small one-page processor, designed to be embedded in some avionic equipment. Only the processor page is included (no memory, I/O, power supply or box).
3. CC-1 and CC-2 are considerably larger scale computers, and are included primarily for comparison purposes.
4. Zilog Z8001, Motorola 68000 and Intel 8086 are current technology, single-chip 16-bit microprocessors.

Failure rates (as well as other operating characteristics) for the first three classes of processors are given in Table 2-5. Table 2-6 lists the failure rates by subassembly level. Failure rates and operating characteristics are also presented for RAM and core memory pages (Table 2-7), microprocessors and LSI support devices (Table 2-8), and power supplies (Table 2-9). A breakdown of each processor's failure rate by subassembly appears in Tables 2-10 through 2-15.

All failure rates presented herein reflect operation in an airborne inhabited fighter environment which typifies F-15 flight operation. (If needed, comparable failure rates for an airborne uninhabited environment can be obtained by multiplying the airborne inhabited rates by a factor of 1.75.) The failure rates are represented by λ and are always expressed in units of failures per million hours. The reciprocal of λ is MTBF (mean time between failures) and is always presented in units of hours.

The fault detection capability percentages appearing in Table 2-5 are primarily derived from predictions and consequently are only estimates of actual system operation. All other data in Table 2-5, except predicted MTBF, reflects the latest available actual system characteristics.

Table 2-8, which addresses microprocessors and LSI devices, contains two sets of failure rates. The first is referred to as "device failure rate" and is the airborne inhabited failure rate for the device by itself. The second column, "device and associated hardware failure rate", also reflects an airborne inhabited environment, but takes into account essential hardware connected with each device. This hardware includes solder, connections, MIBs and capacitors, etc. These failure rates are not measured data, but were developed in accordance with MIL-HDBK-217C.

No detailed F-15 environment reliability predictions were available for the processors profiled in this report. Consequently it was necessary to perform an F-15 environment prediction on each processor. The predictions were done at the subassembly level. The subassembly failure rates listed in Table 2-6 were derived by averaging various processor failure rates normalized to an F-15 environment. Detailed prediction results for each processor appear in Tables 2-10 through 2-15.

The data presented so far relates specifically to the components of the distributed computer network. During the Phase II portion of the study, the plan is to do the reliability analysis on the full avionic suite (as shown in Figure 2-3). In order to do this, reliability data is also needed for the various avionic subsystem equipments. This information was supplied by AFAL, and is shown in Table 2-16. Three MTBF values are given:

1. Field MTBF - Measured data from operational F-15 systems, for the period of January to June, 1979.
2. Specified MTBF - Contractual MTBF requirement.
3. RQT MTBF - Demonstrated MTBF during reliability qualification tests.

TABLE 2-5. PROCESSOR SUMMARY CHART

Processor Name	Physical Size (in)	Weight (lbs)	Performance (KOPS)*	Part Count Total/IC	No. of SRU's	Power Dissipation	Predicted MTBF	Fault Detection Capability
AP-1 (F-15)	7.6 x 12.7 x 15.6 (0.87 ft ³)	36	450	3400/1400	11	200	2,750	>98%
AP-101 (Space Shuttle)	7.6 x 10.0 x 19.6 (0.87 ft ³)	58	550	6340/2120	24	327	1,100	>95%
AP-101C (B-52 G/H)	8 x 10.8 x 20 (1 ft ³)	61	500	3425/1780	22	575	1,420	94.1%
AYK-14 (LAMPS)	7.6 x 10.1 x 19.6 (0.87 ft ³)	51	435	N/A	14	448	2,120	97%
BLP	3.5 x 7.5 x 0.5	-	550	-	1	30	37,037	-
CC-1 (AWACS)	20 x 40 x 73 (33.8 ft ³)	1500	730	N/A/17,300	161	3,859	170	98%
CC-2	20 x 45 x 72	1500	2490	53,425/ 21,950	201	6,363	130	98%

* Performance value is very dependent on instruction mix.

TABLE 2-6. TYPICAL SUBASSEMBLY FAILURE RATES

Package	Typical Logic Page λ^*	Typical I/O Page λ^*	Typical Power Supply λ^*	Typical Memory Page λ^*			Typical PROM Page λ^*	Typical RAM Page λ^*	1553 Page λ^*
				8K (MCM)	16K (DMCM)	32K (QMCM)			
I	23.4	22.2	69.6	38.9	40.4	-	74.9	71.7	-
MCS	24.9	18.8	58.1	-	-	11.7	-	52.9	31 (AF-1010) 37 (AYK-11)

* Failure rates are in failures per million hours.

Note: All failure rates have been normalized to an F-15 operational environment.

TABLE 2-7. MEMORY PAGE FAILURE RATES

Type	Size	λ^*
Core		
QMCM	32K x 18	42
Fabritek	32K x 18	88
Static RAM		
SPS/SED	4K x 34	38
	Control Store	
Working Store	4K x 20	32
Program Store	8K x 36	137
Bulk Store	8K x 36 w/error correction coding	51
Dynamic RAM		
Proteus ASP	16K x 36	55
(Prediction)	16K x 18	28
BLP (C-MUX)	16K x 18	67

* Failure rates are in failures per million hours.

NOTE: All failure rates have been normalized to F-15 operational environment.

TABLE 2-8. MICROPROCESSOR AND LSI DEVICE FAILURE RATES

Device	Device Failure Rate (λ^*)	Device & Associated Hardware Failure Rate (λ^*)	Description
Z8001	0.343	0.526	40-Pin DIP, Microprocessor (310 KOPs)
Z8002	0.392	0.591	48-Pin DIP, Microprocessor
68000	0.610	0.840	64-Pin DIP, Microprocessor (500 KOPs)
8086	0.374	0.557	40-Pin DIP, Microprocessor (210 KOPs)
8251	0.194	0.3536	28-Pin DIP, Programmable Communication Interface
8253	0.178	0.330	24-Pin DIP, Programmable Interval Timer
8255A	0.242	0.425	40-Pin DIP, Programmable Peripheral Interface
8257	0.442	0.625	40-Pin DIP, Programmable DMA Controller
8259	0.174	0.334	28-Pin DIP, Programmable Interrupt Controller

*Failure rates are in failures per million hours.

Note: These failure rates were derived using the techniques established in MIL-HDBK-217C; they are not empirical data.

All failure rates have been normalized to an F-15 operational environment.

TABLE 2-9. POWER SUPPLY CHARACTERISTICS

	Description	Efficiency (%)	λ^*
AP-101C (B-52 G/H)	+5 V at 60.7 A, +5.2 V at 2.4 A +7.4 V at 3.04 A, +12 V at 2.08 A -5 V at 1.66 A	72.1	58
AN/BQQ-5	355 parts, 380 power dissipation	N/A	35
ADTECH AC-DC	5.2 V \pm 0.2% at 10 A max 12 V \pm 0.2% at 5 A max 15 V \pm 0.2% at 4 A max	74 77 78	57 57 57
DC-DC	In Out +12 V 5.25 V 28 V 5.25 V 48 V 5.25 V	62 70 72	59 59 59

*Failure rate is in failures per million hours.

Note: All failure rates have been normalized to an F-15 operational environment.

TABLE 2-10. FAILURE RATE SYNTHESIS FOR AP-1

Page type	Qty	SRU ($4\pi/\text{MCS}$)	λ^* Flight	N λ^*
Logic	5	4π	23.4	117.0
I/O	3		22.2	66.6
Memory	2 (16K)		40.4	80.8
Supplies	1		69.6	69.6
Misc	-		29.8	29.8

λ Total = 363.8
 MTBF = 2,748.8 h
 MTBF = 2,750 h

*Failure rate is in failures per million hours

TABLE 2-11. FAILURE RATE SYNTHESIS FOR AP-101

Page type	Page Qty	SRU ($4\pi/\text{MCS}$)	λ^* Flight	N λ^*
Logic	9	4π	23.4	210.6
I/O	2		22.2	44.4
Memory	10 (16K)		40.4	404.0
	2 PROM		74.9	149.8
Supplies	1		69.6	69.6
Misc	-		33.5	33.5

λ Total = 911.9
 MTBF = 1,096.7 h
 MTBF = 1,100 h

* Failure rate is in failures per million hours

TABLE 2-12. FAILURE RATE SYNTHESIS FOR AP-101C

Page type	Page Qty	SRU		N λ *
		(4 π /MCS	λ * Flight	
Logic	9	MCS	24.9	224.1
I/O**	6		31.0	186.0
	2		18.8	37.6
Memory	4 (32K)		41.7	166.8
Supplies	1		58.1	58.1
Misc	-		34.0	34.0

λ Total = 706.6
 MTBF = 1,415.2 h
 MTBF = 1,420 h

*Failure rate is in failures per million hours.

**For I/O, the 31.0 represents the 1553 page and 18.8 represents the other I/O pages.

TABLE 2-13. FAILURE RATE SYNTHESIS FOR AYK-14

Page Type	Page Qty	SRU		N λ *
		(4 π /MCS)	λ * Flight	
Logic	6	MCS **	24.9	149.4
I/O***	2		37.0	74.0
	2		18.8	37.6
Memory	3 (32K)		41.7	125.1
Supplies	1		58.1	58.1
Misc	-		26.8	26.8

λ Total = 471.0
 MTBF = 2,123.1 h
 MTBF = 2,120 h

*Failure rates are in units of failures per million hours.

**AYK-14 page assemblies have physical characteristics which are similar to MCS and consequently are treated as such.

***AYK-14 has 4 I/O pages 2 of which are 1553 type; 37 is the 1553 page failure rate.

TABLE 2-14. MISSION FAILURE RATE SYNTHESIS FOR CC-1

Page type	Page Qty	SRU (4π /MCS)	λ * Flight	N λ *
Logic	84	4π	23.4	1,965.6
I/O	22		22.2	488.4
Memory	28 (8K)		38.9	1,089.2
	8 RAM		71.7	573.6
	4 PROM		74.9	299.6
Supplies	15		69.6	1,044.0
Misc	-		386.61	386.6

 λ Total = 5,847.0

MTBF = 171.0 h

MTBF = 170 h

*Failure rates are in failures per million hours.

TABLE 2-15. FAILURE RATE SYNTHESIS FOR CC-2

Page Type	Page Qty	SRU (4π /MCS) (QTY/QTY)	λ * Flight 4π /MCS	N λ *
Logic	89	13/76	23.4/24.9	2,196.6
I/O	26	26/0	22.2/-	577.2
Memory	40 (32K)	0/40	-/41.7	1,985.4
	6 RAM	0/6	-/52.9	
Supplies	40	10/30	69.6/58.1	2,439
Misc	-	-	373.9	373.9

 λ Total = 7,572.1

MTBF = 132.1 h

MTBF = 130 h

*Failure rates are failures per million hours.

TABLE 2-16. SELECTED SUBSYSTEM RELIABILITY INFORMATION

Subsystem		Field MTBF	Specified MTBF	RQT MTBF
EAIC	(Electronic Air Inlet Controller)	503(2.4)	1000	1010
FDS	(Flight Director System)	1234(1.5)	1500	4340
ADI	(Attitude Director Indicator)	660(5.7)	1500	4512
ADC	(Air Data Computer)	480(2.3)	800	809
HSI	(Horizontal Situation Indicator)	344(2.5)	680	909
AFCS	(Automatic Flight Control Set)	157(2.3)	475	708
SDRS	(Signal Data Recorder Set)	49(1.2)	570	1789
CC	(Central Computer)	180(2.4)	1000	1426
ICNCP	(Integrated Comm., Nav. and Identification Control Panel)	82(4.5)	980	1218
IFF	(Identification Friend or Foe)	227(2.2)	500	762
IRE	(IFF Reply Evaluator)	553(0.9)	700	722
IMU	(Inertial Measurement Unit)	102(1.5)	--	--
ADF	(Automatic Direction Finder)	2865(0.3)	1000	1434
ILS	(Instrument Landing System)	1794(1.0)	1000	1135
AHRS	(Attitude Heading Reference System)	148(2.3)	750	933
LCG	(Lead Computing Gyro)	350(1.6)	800	1206
RADAR	(Fire Control Radar)	16(1.5)	60	38
VSD	(Vertical Situation Display)	137(1.8)	500	569
HUD	(Head Up Display)	67(-)	225	286
ACS	(Armament Control Set)	264(-)	300	325
IBS	(Interference Blanker Set)	1870(2.1)	1000	1060

() = operating hours/flying hour

2.3 TASKS 1.3 AND 1.5 - ERROR CATALOG

The objectives of this part of the Phase I study are threefold:

1. Identify the various types of errors that can occur within the distributed computing network.
2. Define one or more error checkers capable of detecting each error type.
3. Develop descriptive data about each error checker and its effect when included in the computing network.

The results of this portion of the study are given in the error catalog in paragraph 2.3.2

One of the principle objectives of the Phase II study to follow is to evaluate the cost/effectiveness of these error checkers; i.e., determine which ones are worth including in a computing system and which ones are not.

2.3.1 DEFINITIONS

A fault is a physical defect in a circuit which causes, either temporarily or permanently, some form of out-of-specification performance. If the fault is a permanent defect, it is termed solid. If the fault occurs only once, or for a very short period of time, it is termed transient. (A transient fault might recur from time to time, in which case it would be considered intermittent; but that normally does not affect the way the fault is handled.) An error is a symptom of such a fault. Hardware and software techniques for detecting errors in the computer network are called error checkers, and are the principle subject of this portion of the study. An on-line checker is implemented in hardware and has a duty cycle of 100%. An off-line checker is implemented in software (a self-test program or a diagnostic program, or a check built into an operational program) and has a duty cycle dependent on the frequency of test execution. Off-line checkers are in general not effective against transient errors.

As shown previously in Figure 1-2, each processing element in the computer network can be broken down into four major functions; namely, CPU, Memory, BIU, and I/O interface. The catalog of error types and error checkers given in Paragraph 2.3.2 is developed separately for each major function. To clarify this breakdown, Table 2-17 gives these major functions, and the subfunctions that are typically part of them.

TABLE 2-17. MAJOR FUNCTION BREAKDOWN

<u>CPU</u>	<u>Memory</u>
ALU	Memory array
Local Store (registers)	Address decode
Timing and controls	Data select
Interrupts	X drivers } For core
Data Flow	Y drivers } memories
Memory address generation	Sense amps }
Control store	Controls
BIU control interface	Memory bus interface
	Memory access control
<u>BIU</u>	<u>I/O</u>
Simple ALU	Data flow
Control store	Device address generation
Data bus interface	Drivers and receivers
Buffer storage	Timing and controls
Processor interface	Serial/parallel conversion
Timing and controls	Device interface
Serial/parallel conversion	Memory bus interface
	CPU interrupt controls
	Buffer storage
	Memory access control

Once an error is detected, there are two kinds of actions that might be taken. Retry is an attempt to re-execute the specific function or process that was going on when the error occurred. Retry normally happens automatically and immediately, with no software intervention. If retry is successful, the error is assumed to have been transient, and system operation proceeds just as if no error had occurred. If the retry is not successful, the error is assumed to be solid, and other recovery action is necessary.

Whenever a solid error occurs, it can usually be isolated to one of the major functions in one of the processing elements in the computer network. Reconfiguration is the process of restructuring the computer network hardware so that it will operate without the failing major function. This normally requires special hardware switching facilities to allow this to happen, and involves the use of spare modules. Operation of the application programs is usually interrupted while this takes place, and software support is needed to decide upon and invoke the proper reconfiguration actions, and to reorganize the executive and operational programs to allow system operation to resume with the modified hardware configuration.

2.3.2 ERROR CATALOG

The catalog of error types and error checkers is presented in a series of eight tables, one table for each error catalog of the four major functions outlined in Table 2-17, and one table for each set of descriptions of the various error checkers.

The following is a description of each of the columns found in the error catalog tables, with information on how to interpret the data presented there.

1. ERROR - This is a short phrase identifying the particular error type that could occur within the major function
2. CHECKER - One or more error checkers are identified for each of the error types. A separate description is provided for each one.
3. CHECKER NUMBER - This is an encoded numbering scheme for identifying each unique error checker. This is a six-digit number, encoded as follows:

X1, X2, X3, X4, X5, X6

X1 - Checker type

1 - On-line checker (BITE)

2 - Off-line checker (Software)

X2 - To be defined

X3, X4 - Function Test

01 - 99

01 - CPU

02 - Memory

03 - I/O
04 - System Bus
05 - Power

X5, X6 - Subfunction Test
00 - 99

4. % DETECT - One measure of the effectiveness of an error checker is the percentage of error occurrences it can detect. The value given in the table represents the percentage of that particular type of error which the checker will detect. The values given are engineering estimates based on two sources: data collected on a large number of IBM's commercial computers (System/360 and System/370); and detailed testability analysis of three of IBM's military computers. Note that the values in the table are not additive; that is, if several of the listed error checkers are included in one major function, an analysis is required to determine what percentage of all the errors that could occur would in fact be detected by that set of checkers.
5. CHECKER COVERAGE % - Another measure of the effectiveness of an error checker is the amount of hardware it is able to check. The coverage value given in the table is the percentage of the total hardware required to implement the major function, which is checked by the particular checker. The values given in the tables are engineering estimates based on the detailed testability analyses mentioned above. For any given configuration the actual values will be design and technology dependent. Since some of the checkers will overlap on the hardware they check, these coverage values cannot simply be summed up, either.
6. RECOVERY ACTION - This category was intended to suggest alternative recovery actions which might be taken whenever a solid error was detected by the particular checker. However, it has turned out that the options are limited. There are two limiting conditions:
 - a. If appropriate redundant modules and hardware switching are not available, recovery is not achievable.
 - b. If sufficient additional hardware is provided, any error condition can be recovered from.Consequently, this column is encoded as follows:
 - R - Reconfiguration is possible, with proper redundant hardware modules and software support.
 - X - Recovery is not required. (This is applied to the use of error correcting codes, where the first failure is automatically and continuously corrected.)
 - N - No recovery is possible. This is only used in a few cases, where the corrective hardware would have to be implemented someplace other than within the distributed computer network.
7. RETRY - This column indicates whether retry might be used to get around transient errors. There would be of course a

hardware cost associated with such a retry. The column is encoded as follows:

- H - Retry could be implemented with suitable hardware.
- S - Retry could be done in software.
- X - Retry is not required.
- N - Retry is not achievable.

8. COST - This column is intended to give a rough relative cost of the different error checkers. There appears to be no really precise way to indicate this, since every measure is dependent upon technology, machine size, and machine design. The value in the table is a percentage of the hardware used to implement the complete major function. For example, the first entry in Table 2-18 shows that the parity checker in main store (i.e., the parity bit in every word of memory, plus the circuits to compute and check parity whenever needed, plus the circuits required to notify the CPU when an error is detected) would require an amount of hardware equivalent to 6% of the total hardware used to implement the full memory system. A footnote with each table indicates the full hardware size, upon which the percentage is based.

With these descriptions in mind, the error catalog follows.

2.3.2.1 Memory Function

The error catalog for the memory function is listed in Table 2-18. The error checker descriptions are given in Table 2-19.

TABLE 2-18. MEMORY FUNCTION ERROR CATALOG

Error	Checker	Checker Number	Detect	Checker Cover %	Recovery Action	Retry	Cost*
Single data bit error	Parity	100200	99.0%	72.0%	R	H	6
	Parity with ECC	100201	99.5%	72.0%	X	X	26
	Checksum	200201	95.0%		R	S	-
Multi data bit error	Parity	100200	50.0%	72.0%	R	H	6
	Parity with ECC	100201	50.0%	72.0%	R	H	26
	Checksum	200201	40.0%		R	S	-
Single bit addressing error	Parity (address)	100202	87.0%	12.0%	R	H	0.5
	Parity with ECC (addr)	100202	87.0%	12.0%	X	X	1
	Store protect	100203	40.0%	36.0%	R	H	6
Multiple bit address error	Parity (address)	100202	43.0%	12.0%	R	H	0.5
	Parity with ECC (addr)	100202	43.0%	12.0%	R	H	1
	Store protect	100203	40.0%	36.0%	R	H	6
Memory bus I/F error (Single bit)	Memory bus time-out	100204	8.0%	24.0%	R	H	1.5
	Interface parity	100205	99.0%	12.0%	R	H	0.5
	I/F parity with ECC	100206	99.5%	12.0%	X	X	1
Memory bus I/F error (Multi bit)	Memory bus time-out	100204	8.0%	24.0%	R	H	1.5
	I/F parity	100205	50.0%	12.0%	R	H	0.5
	I/F parity with ECC	100206	50.0%	12.0%	R	H	1

*Cost basis is eight memory pages plus one memory controller page.

TABLE 2-19. MEMORY FUNCTION ERROR CHECKER DESCRIPTION

Checker Name	Description
Parity (100200)	This checker is the presently used parity checker common to many memories, where one parity bit is provided for each word (or halfword, or byte) in the memory. It is capable of detecting single bit picks and drops. There is no error correction capability in this type of checker. Certain types of multiple bit failures may be detected, but they must be of such a nature that the net effect will be to invert the parity bit.
Parity with ECC (100201)	Parity with error correction can be implemented for single bit failures, by adding an error correcting code to each word in memory. In this way it is possible for a processor to function with single bit picks or drops in every word of memory. The error-correcting code can also detect double-bit failures.
Parity (Address) (100202)	The parity checker for address bit failures is defined separately, since parity checking of the address would only occur on the memory bus, while parity checking of data could occur either on the bus or in memory, or both. Note that a single parity bit could be used on the bus to check both address and data. An error correcting code could also be included with the address.
Store Protect (100203)	Store protect is a common checker seen in many memories. It provides detection of addressing failures (regardless of the number of faulty bits) if and only if the failure causes a store operation to be attempted into a protected location. Consequently, it is not very effective as an address checker.
Memory Bus Time-Out (100204)	If the expected response is not received from the addressed unit within a specified time, an error is generated. This checker will detect control type faults on the memory interface. No data fault detection is provided, so for a parallel bus the coverage is low (i.e., there would be few control lines relative to data lines).
Interface Parity (100205)	See "Parity, 100200" above.
I/F Parity with ECC (100206)	See "Parity with ECC, 100201" above.
Checksum (200201)	This checker is a software routine written by the programmer. It does an arithmetic sum (ignoring overflows) of any part or all of memory. Then a compare is made to a known value to determine if any locations have been altered. (Note: The checksum constant must be generated sometime previous to performing the check.) The coverage percentage cannot be specified, since it depends strictly on how many words in memory happen to be included in a particular checksum function.

2.3.2.2. I/O Function

The error catalog for the I/O function is listed in Table 2-20.
The error checker descriptions are given in Table 2-21.

TABLE 2-20. I/O FUNCTION ERROR CATALOG

Error	Checker	Checker Number	% Detect	Checker Cover %	Recovery Action	Retry	Cost *
Single bit error in XMIT	I/O I/F parity	100300	99.0%	40.0%	R	H, S	6
	I/O I/F parity with ECC	100301	99.5%	40.0%	X	X	18
Multi bit error in XMIT	I/O I/F parity	100300	50.0%	40.0%	R	H, S	6
	I/O I/F parity with ECC	100301	50.0%	40.0%	R	H, S	18
Stack DLE DO	Redundant handshake	100302	85.0%	40.0%	N	H	2
False interrupt request	Invalid int. code	100303	90.0%	20.0%	N	H	2
Memory Address error (Single bit)	Store protect	100203	45.0%	65.0%	R	H, S	6
	Mem I/F parity	100304	99.0%	20.0%	R	H, S	0.5
(Multi bit)	Mem I/F parity with ECC	100305	99.5%	20.0%	X	X	1
	Mem I/F parity	100304	50.0%	20.0%	R	H, S	0.5
	Mem I/F parity with ECC	100305	50.0%	20.0%	R	H	1
I/O I/F error	I/O I/F time-out	100307	90.0%	40.0%	N	H, S	1.5

* Cost basis is a one-page I/O controller.

TABLE 2-21. I/O FUNCTION ERROR CHECKER DESCRIPTIONS

Checker Name	Description
I/O I/F Parity (100300)	I/O Interface parity checking is designed to detect single bit failures in transmission of data (or device address or status information) to or from an I/O device.
I/O I/F Parity with ECC (100301)	An error correcting code can be included in the I/O interface which would automatically correct single bit errors, and detect double bit errors.
Redundant Handshake (100302)	Detecting a discrete input or discrete output that is stuck (either on or off) requires extra controls - either additional lines or additional sequencing - to determine if the discrete signal gets reset after it has been accepted.
Invalid Interrupt Code (100303)	When a device sends an interrupt to the processor, it is usually accompanied by an interrupt code to indicate the cause or the nature of the interrupt. An invalid code value is an indication of a false or incorrect interrupt signal.
Mem I/F Parity (100304) Mem I/F Parity with ECC (100305) Store Protect (100203)	These checks on the I/O interface to the memory are the same as those defined above for the memory function

2.3.2.3 System Bus Function

The error catalog for the system bus function is listed in Table 2-22. The error checker descriptions are given in Table 2-23.

TABLE 2-22. SYSTEM BUS FUNCTION ERROR CATALOG

Error	Checker	Checker Number	% Detect	Checker Cover %	Recovery Action	Retry	Cost*
Single Bit error in xmit	Bus parity	100400	99.0%	56.0%	R	H, S	0.5
	Bus parity with ECC	100401	99.5%	56.0%	X	X	1
Multi bit error in xmit	Bus parity	100400	50.0%	56.0%	R	H, S	0.5
	Bus parity with ECC	100401	50.0%	56.0%	R	H, S	1
Bus hang-up	Bus time-out	100402	30.0%	42.0%	R	H	1.5
Illegal unit address	Bus protocol	100403	45.0%	14.0%	R	H	2
	Addr parity check	100406	98.0%	57.0%	R	H	0.5
	Acknowledge check	100407	45.0%	48.0%	R	H	0.5
	Bus time-out	100402	30.0%	50.0%	R	H	1.5
Illegal sequencing	Addr validity chk	100409	98.0%	45.0%	R	H	1
	Addr validity chk	200400	35.0%	45.0%	R	S	-
	Bus protocol	100403	55.0%	14.0%	R	H	2
	Bus time-out	100402	45.0%	42.0%	R	H	1.5
Priority error	Fault tolerant ckt	100404	85.0%	42.0%	X	H	5
Illegal control code	Decode chk	100410	40.0%	30.0%	R	S	2
Wrong word count	Word count chk	100411	45.0%	30.0%	R	H	1
	Word count chk	200401	15.0%	30.0%	R	S	-
Wrong bit count	Bit count chk	100412	35.0%	25.0%	R	H	1
No bus response	Bus time-out	100402	35.0%	80.0%	R	H	1.5
Controller inoperative	No activity time-out	100408	50.0%	55.0%	R	H, S	1.5
Controller hangs bus	BIU time-out	100413	65.0%	85.0%	R	S	1.5
Terminal always busy	Retry count	100414	75.0%	60.0%	R	H	2
	Retry count	200402	75.0%	60.0%	R	S	-
No terminal response	Bus time-out	100402	65.0%	60.0%	R	H, S	1.5

*Cost basis is a one-page BIU.

TABLE 2-23. SYSTEM BUS FUNCTION ERROR CHECKER DESCRIPTIONS

Checker Name	Description
Bus Parity (100400) Bus Parity with ECC (100401)	Parity checking on the system bus is the same technique as described for the memory bus
Bus Time-Out (100402)	If the expected response is not received from the addressed unit within a specified time, an error is generated. This checker will detect control type faults on the system bus, as well as certain faults in the BIUs attached to the bus.
Bus Protocol (100403)	Careful protocol design will yield detectability of illegal device addresses and various sequencing errors. The levels of detection and coverage that can be achieved are very dependent on the characteristics of the basic protocol.
Fault Tolerant Priority (100404)	Detecting errors in message priority interpretation could be very difficult to implement, depending on the basic bus protocol. It would most likely involve some redundancy, and special microcode.
Addr. Parity Check (100406)	This checker provides good detection for circuitry in the bus address generator. Implementation is similar to parity checkers in memory and I/O functions.

TABLE 2-23. SYSTEM BUS FUNCTION ERROR CHECKER DESCRIPTIONS
(Cont'd)

Checker Name	Description
Acknowledge Check (100407)	An acknowledge from the wrong unit is an indication of a message addressing error.
No Activity Time-Out (100408)	The assumption is made that activity from every unit will always occur in a given time period. If this expected activity does not occur an error signal is generated.
Addr. Validity Check (100409)	Checking a message address before it is put on the bus detects only software errors. The check must be made after it is put on the bus.
Decode Check (100410)	Provides error detection for bus function decoders. They provide error detection earlier than some other checkers. This also must be checked on the bus, or by the receiver of the message.
Word Count Check (100411, 200401)	This checker verifies that the correct number of words were received (and/or sent). This can also be checked by software.
Bit Count Check (100412)	This checks the proper number of bits per word, and will also detect some sequencing difficulties, depending on the protocol. It is most useful on serial busses.
BIU Time-Out (100413)	Any BIU can observe the traffic on the bus, and determine if one BIU has failed to release the bus.
Retry Count (100414, 200402)	Failure of a BIU to become nonbusy can be detected by either hardware or software, by retrying a message some selected number of times.

2.3.2.4 CPU Function

The error catalog for the CPU function is listed in Table 2-24. The error checker descriptions are given in Table 2-25.

TABLE 2-24. CPU FUNCTION ERROR CATALOG

Error	Checker	Checker Number	% Detect	Checker Cover %	Recovery Action	Retry	Cost*
Microstore bit failure	Microstore parity	100101	98.0%	16.0%	R	H	1
	Microstore parity with ECC	100102	99.5%	32.0%	X	X	6
ALU error	Parity predict on ALU	100103	98.0%	48.0%	R	H	20
(Single bit)	Residue checking	100110	99.5%	48.0%	R	H	2
(Multi bit)	Residue checking	100110	93.0%	48.0%	R	H	2
LS addressing error	LS addr parity	100104	87.9%	32.0%	R	H	1
	LS addr par with ECC	100105	87.5%	32.0%	X	X	2
Decoding error	Illegal OP codes	100106	35.0%	32.0%	R	H	0.5
Illegal instruction	Illegal OP codes	100106	95.0%	32.0%	R	H	0.5
LS bit failure	LS parity	100107	98.0%	32.0%	R	H	2
	LS parity with ECC	100108	99.5%	32.0%	X	X	4
CPU hang-up	GO/NO GO timer	100109	65.0%	64.0%	R	N	2
	Activity timer	100111	95.0%	64.0%	R	N	2

*Cost basis is a five-page CPU.

TABLE 2-25. CPU FUNCTION ERROR CHECKER DESCRIPTIONS

Checker Name	Description
Memory Bus Time-Out (100100)	This checker is used to detect gross timing errors in the memory function.
Microstore Parity (100101) Microstore Parity with ECC (100112)	Parity checking in microstore is the same technique as described for memory.
Parity Predict on ALU (100103)	
LS Addr Parity (100104) LS Addr Par with ECC (100105)	Parity checking on the local store (LS) address is the same technique as described for the memory address. Note that the CPU register set is the principle content of the local store. On some machines this may not be implemented as a storage array, which would affect the checker cost.
Illegal OP Codes (100106)	
LS Parity (100107) LS Parity with ECC (100108)	Parity checking in local store is the same technique as described for memory.
CO/NOGO Timer (100109)	
Residue Checking (100110)	This checker is used as a gross detection of CPU failures that result in incorrect or unexpected instruction sequences. It catches a large class of otherwise unchecked errors. However, there is potentially a very long delay between the occurrence of the fault and the detection of the error, and there is literally no indication of what caused the error. Generally, there must be a little software support for the use of this timer.
Activity Timer (100111)	This checker is very effective on single bit failures. Multi-bit detection is high if residue 15 checking is used. A residue 3 check (which requires two redundant check bits) provides 67% coverage, and residue 15 check (four bits) provides 93% coverage.
	This is a simple timer check which, by monitoring some standard CPU control point (such as I-fetch, OP decode, or ENDOP), ensures that the CPU is continuing to execute instructions.

2.4 TASK 1.4 - BUS CONTROL PROCEDURES

The objective of this portion of the study was to select three system bus candidates for more detailed study and evaluation in Phase II. As a result of the selection process, eight possibilities were considered, and actually four are recommended for further study.

This subsection describes the selection process and the eight buses considered, and the resulting evaluation that lead to the recommendation.

2.4.1 SELECTION APPROACH

The selection approach is divided into three steps. In the first step, a set of criteria is developed against which candidate data buses are evaluated. These criteria are general and tend to define bus characteristics that would be required in a distributed, fault tolerant processor network for an avionics application. These criteria are discussed in paragraph 2.4.2.

The second step in the selection procedure is the definition of the set of data buses that will be evaluated against the selection criteria. Paragraph 2.4.3 describes a set of eight distributed data buses. The buses are presented in order of increasing complexity and sophistication; ranging from MIL-STD-1553B to the overlapped control bus used in IBM's Future Signal Processor. These buses represent a reasonable extrapolation of the state-of-the-art in data bus design.

In the final step, described in paragraph 2.4.4, an evaluation matrix is developed in which the eight data buses are rated against the selection criteria. The four buses that obtain the highest ratings are recommended for future study and more rigorous evaluation.

This selection procedure uses qualitative measures based on engineering judgment to narrow the field to four buses that will later be evaluated quantitatively. By testing the inherent system driven requirements against the reality of some existing implementations, it is expected that realistic, implementable buses were selected for further evaluation.

2.4.2 SELECTION CRITERIA

The distributed data bus is perhaps the central component and largest potential bottleneck around which the entire design of a distributed processor network revolves. For the purposes of this task, a distributed network of processors is being configured as a fault tolerant avionics system. Fault tolerance implies the absence of any single point failures, and automatic reconfiguration in case of failure. This

overall fault tolerance philosophy, then, can be broken into a set of specific data bus requirements. This section presents these criteria.

These criteria are grouped into four major areas of concern:

1. System organization
2. Bus access method
3. Message structure
4. Electrical and technology.

Each of these areas is discussed in detail in the following paragraphs.

2.4.2.1 System Organization Criteria

At the highest system level a data bus that must support a fault tolerant, reconfigurable network must have the following characteristics:

1. Decentralized Control - The bus access and control method must not be centralized. A single failure in one bus interface unit (BIU) should not render the bus inoperable. If decentralized control is not used, then the central bus allocation and control logic must be duplicated.
2. Error Checking - The bus must be able to survive an intermittent failure. Automatic command and message retry on the occurrence of an error should harden the system to intermittent, noise type, failures. A hard, solid failure in the data bus should be quickly detected, and immediate reconfiguration switchover to an alternate bus should be initiated.
3. Efficient Protocol - The message protocol should be straightforward and not tie the bus up in unnecessary handshaking.
4. Closed-Loop Control - Every transmission should receive a positive acknowledge. In this manner a sequence can be aborted early due to a transmission error or it can be retried a small part at a time. Also, an error in device selection can be discriminated from command and data transmission errors. Tight control enhances traceability and quick error checking and recoverability.
5. Data Transfer Busy Hold-off - "Ready or not, here it comes" data transfer message protocols should be avoided. A hold-off mechanism should be used that allows the data sink to delay the data transfer until it is ready to receive data. Sometimes a processor-controlled data sink must process an interrupt to determine the storage location of an incoming message. A hold-off mechanism allows the data sink to delay transfer of the message until the interrupt is processed or storage is available. Absence of a data transfer busy mechanism results in large hardware data buffers and correspondingly expensive BIUs.

6. Sufficient Data Throughput - After derating for bus overhead, such as bus allocation and message protocol, the data throughput should be sufficient to handle a representative avionics mission load. Specifically, the data bus should be capable of sustaining a minimum load of 5,824 sixteen-bit words/second (Reference 3). Sufficient additional bandwidth should be included to allow system growth (attachment of new devices, or more processors in the network).
7. Low BIU Cost/Complexity - The distributed bus and, consequently, the overall BIU design should not become so complex that the BIU hardware becomes a significant portion of the processor/BIU pair. Even if a "perfect" data bus could be developed, if the resulting BIU were so complex as to be impractical, then the bus could not be used in the weight and volume critical avionics environment. Data bus features have to be traded against the cost of hardware implementation.

2.4.2.2 Bus Access Requirements

To support reconfiguration, each processor in the network must be able to assume every other processor's tasks. In this manner, a failure in one processor can be taken over by another processor in the network. This philosophy includes overall bus control. To prevent a single BIU/processor failure from taking out the bus, a distributed bus access method is required, which implies the following criteria:

1. Straightforward Allocation Method - The method for allocating the bus control in a distributed decentralized manner should be as simple and as straightforward as possible. Contention resolution should not depend on the detection of data conflicts or loss of data integrity as an allocation method. Loss of data integrity can cause misinterpreted system control commands and possible loss of system control. Control hand-off should be systematic and well ordered.
2. Easily Monitorable - A failure in the bus allocation mechanism should be quickly detected and immediate recovery initiated. This is usually accomplished by a monitor BIU. The bus allocation method should be straightforward so as to simplify the amount of hardware required in the monitor function. The monitor function logic must be duplicated in each and every BIU on the bus, so that any BIU can become the monitor.

2.4.2.3 Message Structure Requirements

The overall message structure should support a system in which any task can reside in any processor at any time. The command, address, and data block structure should allow sufficient flexibility to handle any possible task or the future expansion of the system to new tasks, as follows:

1. Expandable Command/Status Architecture - The command and status field definition should not be constraining. Either the command and status fields should be large, or provision should be included in the message structure to allow expansion of command length and status data blocks.
2. Expandable Addressing - In the same manner, a provision should be included in the message structure to select an expanded BIU address.
3. Broadcast Mode - A system mode should be included in which messages can be transmitted to all BIUs on the bus simultaneously.
4. Associative Addressing - System level organization is by task. Since any task can reside in any processor, then addressing of messages should be by task rather than physical BIU address. Each BIU should contain appropriate logic to decode a message directed at a task currently configured in its processing element.
5. Block Transfers - The bus should have a block transfer mode. Variable length data blocks are preferred.

2.4.2.4 Electrical and Technology Associated Criteria

Certain electrical characteristics, such as the number of bus lines, directly affect data bus reliability. Other characteristics such as the ability to operate on long lines with a minimum of clock skew are required by the avionics environment. The following list is a set of electrical and technology associated requirements that seem to be a natural outgrowth of a high reliability avionics data bus:

1. High Noise Immunity - The logic family should have high noise immunity. System drivers, receivers and grounding philosophy should not degrade noise margins.
2. Minimum Interface Lines - "Bus lines require cable drivers, receivers, and connectors, all of which tend to be costly compared to logic. Connectors occupy a significant amount of physical space and are also among the least reliable components in the system. Also, system noise is increased as the

number of switching lines is increased". (Reference 4). The system throughput must be balanced against the cost and reliability of multiple interface lines to achieve a minimum number of lines for a given data rate.

3. Low Signaling Rate - High-speed devices will trigger or threshold on smaller amounts of electrical energy than low-speed devices. As transmission speed increases, proper termination, impedance match, unit-to-unit ground shift, and crosstalk become more critical; consequently, high-speed devices are much more sensitive to unwanted system noise. The system throughput should be balanced against the higher technical risk of a fast signaling rate to achieve the lowest possible rate for a given throughput.
4. Independence From Ground Shifts - The data bus drivers and receivers should isolate the data from the effects of unit-to-unit power supply and ground variations, especially instantaneous AC variations. A common technique in use today is differential (double-ended) drivers and receivers.
5. Ability to Operate on Long Lines Independent of Clock to Data Skew - Quite often units installed on aircraft are separated by relatively long distances. Due to variations in cable manufacture and physical placement within the aircraft, it is possible to develop a time skew between the bus data and its associated clock. At long distances or high data rates this skew can be significant. An avionics bus must be designed so as to minimize the chance for data to clock skew. Manchester encoding is often used to embed the clock in the data stream and, consequently, eliminate skew.
6. AC Coupled - All of the buses in this report are multiparty. If one driver fails to a hard 1 or 0, it cannot be allowed to render the bus inoperative. The most common technique is to AC couple the bus. Another technique involves the use of isolation networks between DC coupled BIUs and the bus. The isolation networks can then be used to degate a defective DC coupled driver from the bus.

2.4.3 EVALUATION DATA BUSES

This section briefly describes eight data buses that are suitable for avionics applications. Table 2-26 summarizes the current system applications of these eight buses and the sources of the associated bus protocols. The buses are presented in order of increasing complexity and sophistication. The first three are those suggested in the original statement of work (Reference 1). Certain buses introduce new techniques and concepts that are modified and used again in the remaining bus descriptions that follow. Two new techniques, in particular, are described in this section: overlay priority and overlapped control.

Overlay, binary, bit-for-bit priority is first described in the tag encoded, distributed, parallel, selector channel (Bus #4). In that implementation, the priority scheme is implemented in parallel. All of the data buses that follow the initial priority resolution description for the parallel, selector channel use either a serial NRZ version of this priority scheme, or a Manchester serial version of this scheme. The detailed description of the priority mechanism is presented once in Section 2.4.3.4. Later sections describe only the modifications to the basic technique. This technique is novel since: 2^N way priority can be resolved in N serial priority frames, the mechanism is distributed, and the technique does not result in data collisions.

TABLE 2-26. EVALUATION DATA BUSES

Data Bus	Type	Source of Bus Protocol
1. Stationary Master	Manchester Serial	AASMMA Report(Ref 5) MIL-STD-1553B (Ref 6)
2. Nonstationary Master	Manchester Serial	AASMMA Report
3. Contention	Manchester Serial	AASMMA Report
4. Tag Encoded Parallel, Selector	Byte parallel DC interlocked	Internal IBM bus adapted from System/370 I/O protocol
5. Tag Encoded Serial, Selector	Manchester Serial	Internal IBM bus adapted from System/370 I/O protocol
6. MIL-STD-1533B with Overlay Priority	Manchester Serial	MIL-STD-1553B with distributed priority
7. Overlapped Control Parallel	Overlapped bus NRZ serial	IBM Future Signal Processor system bus
8. Overlapped Control Serial	Overlapped bus 16-bit parallel	Serial modification of IBM Future Signal Processor bus

Two distributed data systems are presented that use overlapped control. Overlap has long been used to increase processor throughput. This technique, along with overlay priority, was developed for a distributed data bus used in IBM's Future Signal Processor (FSP) program. The entire FSP data bus is described in subparagraph 2.4.3.7. A serial version of the overlapped control, distributed bus is presented in subparagraph 2.4.3.8.

Each data bus is described using a standard format. This standard format should aid in assessing the pros and cons of the various techniques. The following standard characteristics are described for the eight data buses:

1. Brief description of the bus
2. Number of lines
3. Signaling rate
4. Data transfer rate
5. Maximum length
6. Electrical terminal attachment method
7. Error checking
8. Message formats
9. Bus access method
10. Typical write message exchange
11. BIU cost and complexity

2.4.3.1 Stationary Master (MIL-STD-1553B) Data Bus

Bus Description: Centralized control, demand/response, Manchester-encoded, serial bus with broadcast mode.

Number of Lines: One line.

Signaling Rate: 2 MHz.

Data Rate: 0.8 Mb/s (without overhead).

Length: 100 to 300 feet

Terminal Attachment: AC coupled via coupling transformers.

Error Checking: Parity, invalid Manchester, invalid command, message retry.

Message Formats:

20-bit command word

(3)	(5)	(1)	(5)	(5)	(1)
Sync	Terminal Address	T/R	Subadr Mode	Wd count/ Mode code	P

20-bit data word

(3)	(16)	(1)
Sync	Data	P

20-bit status word

(3)	(5)	(11)	(1)
Sync	Terminal Address	Status	P

Bus Access Method: Centralized controller initiates all transfers. Remote terminals can pass data to central controller by setting service request bit in the status word and waiting for the central controller to poll the terminal status.

Sample Write Message Protocol: Central controller initiates transfer with a command word indicating the receiving terminal address and the word count. The command word is immediately followed by the data words. The receiving remote terminal then ends the write sequence by transmitting a status word, indicating any errors incurred during the data transfer.

BIU Complexity Cost: The AC coupling transformers and the Manchester encoder/decoder logic require a moderate amount of hardware to implement an otherwise simple interface.

2.4.3.2 Nonstationary Master

Bus Description: Centralized control, demand/response, Manchester-encoded, serial bus. Bus control is passed from terminal to terminal. The next controller is established by polling the message priority of all remote terminals. The terminal having the highest priority becomes the next bus controller.

Number of Lines: One line.

Signaling Rate: 2 MHz.

Data Rate: 0.8 Mb/s (without overhead).

Length: 100 to 300 feet.

Terminal Attachment: AC coupled via coupling transformers.

Error Checking: Parity, invalid Manchester, invalid command, message retry. In addition, one bus interface unit BIU is designated as system monitor. The monitor BIU verifies all bus control polling sequences, and is responsible for recovering the system in the event that the bus control polling sequence fails.

Message Formats:

20-bit command word	}	(All identical to stationary master).
20-bit data word		
20-bit status word		

To support bus allocation, mode (00000) is reserved for the allocate bus control command, and status word bit 19 is reserved for accept bus control. A new 17-bit polling request message is concatenated onto the 20-bit status word. This message contains the bus priority requirements for the remote terminal.

37-bit priority polling word



Bus Access Method: Remote terminal obtains bus by transmitting its message priority with the status word. Current bus controller polls all message priorities and assigns next bus control to remote terminal with the highest priority. Polling results in high overhead and a subsequent decrease in effective data rate.

Sample Write Message Protocol: (Identical to stationary master)

BIU Complexity/Cost: If the allocation polling is done in software, the BIU complexity is the same as stationary master. However, implementing the bus allocation in hardware will result in a complex BIU. The monitor function will also result in complex hardware.

2.4.3.3 Contention

Bus Description: Distributed control, Manchester-encoded, serial bus. Bus allocation is distributed and the bus is contended for and allocated after each transaction. Allocation method can result in data collisions, which in turn cause incomplete messages. Colliding messages must be retransmitted.

Number of Lines: One line.

Signaling Rate: 2 MHz.

Data Rate: 0.8 Mb/s (without overhead).

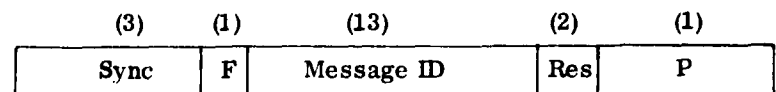
Length: 100 to 300 feet.

Terminal Attachment: AC coupled via coupling transformers.

Error Checking: Parity, invalid Manchester, no response, invalid response. Deferred retry - that is: message is rescheduled and bus is contended for again to retransmit erroneous message. System monitor required to check and recover bus allocation.

Message Formats:

20-bit message ID



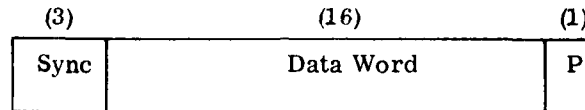
↑ Message ID flag

1 = Message ID word

0 = Request ID word

↑ Reserved bits

20-bit data word



Bus Access Method: A remote terminal that wishes to gain the bus selects a random number representing the number of bus quiet intervals its BIU must wait before attempting transmission. The range of the random number is proportional to the device priority. Devices with higher priorities select random numbers from a smaller delay range than devices with lower priorities. When the current transmission completes, the BIU times out the delay wait number and begins message transmission. The BIU monitors the bus as the message is transmitted and verifies that the data on the bus agrees bit-for-bit with the message. If a miscompare is detected, then two BIU messages have collided. In the event of collision, both BIUs reschedule the messages involved by selecting new random delay wait numbers. If the first 20-bit word is transmitted successfully, then the BIU has gained access to the bus for the remainder of the message.

Sample Write Message Protocol: A BIU that has a write message first gains access to the bus as described above. The first 20-bit word of the write message is the message ID. The message ID contains all message information required by the data sink to receive and store the data, or look up the required information. All data sinks on the bus decode the message ID to determine if the data is for them. The source immediately follows the message ID with the data, and the sequence is completed. No status is returned by the sink. The source must gain access to the bus and interrogate the sink, via a different message ID, to obtain status.

BIU Complexity Cost: The allocation method hardware implementation will result in a complex BIU. The monitor function will further complicate the hardware design and add cost.

2.1.3.4 Distributed, Tag-Encoded, Parallel, Selector Channel

Bus Description: Distributed control, parallel, byte wide data bus. All transfers are DC, request/acknowledge, interlocked using the Transfer In/Transfer Out control lines. The device using the bus is responsible for the next allocation. Allocation takes one bus cycle and does not result in data collisions. Similar to IBM System/370 I/O channels.

Number of Lines: 22 lines total
10 lines - Bus In
10 lines - Bus Out
2 lines - Transfer In/Transfer Out
(See Figure 2-6)

Signaling Rate: 1 MHz.

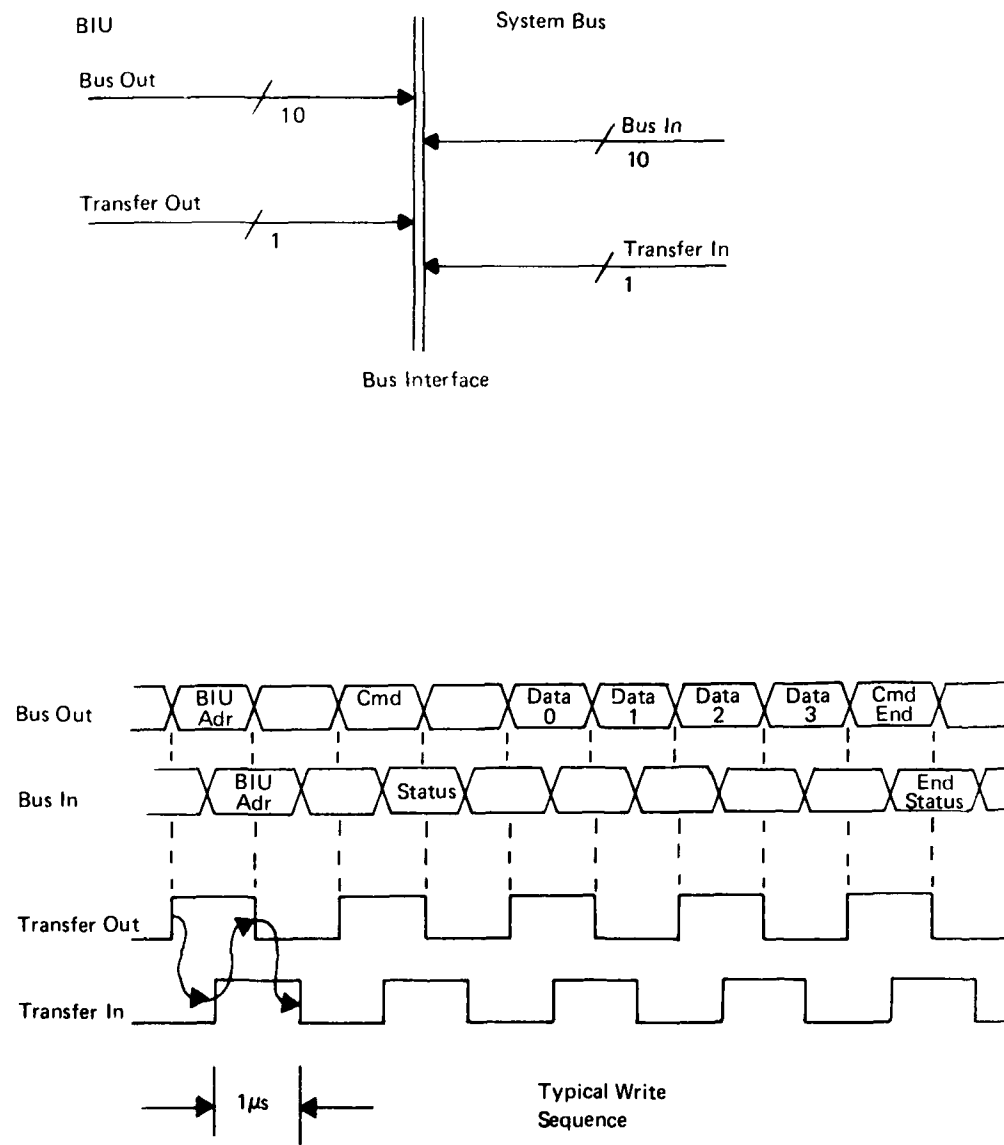


Figure 2-6. Distributed, Tag Encoded, Selector Channel

Data Rate: 8.0 Mb/s (without overhead).

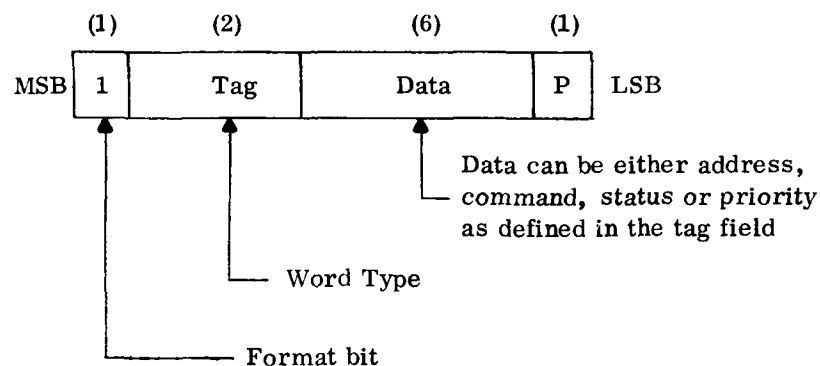
Length: 200 feet (data rate degrades beyond 200 feet).

Terminal Attachment: DC coupled via open collector or tri-state logic drivers.

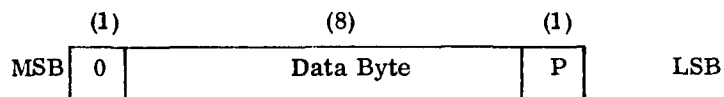
Error Checking: Bus parity, invalid command. Both command and data retry. System monitor required to check and recover system bus allocation failures.

Message Formats: Formats are identical for the 10 data lines of Bus In or Bus Out.

Command/Address/Status/Format



Data Format



Bus Access Method: The last BIU that uses the bus is responsible for the next bus allocation. To accomplish allocation, the last user gates his priority number on the least significant 6 bits of Bus Out, identifies a priority word in the tag field, and activates the Transfer Out interface line. Other BIUs decode the priority code in the Bus Out tag field and respond by also gating their priority numbers onto Bus Out. The highest priority code is all zeros and the lowest priority code is all ones. The lowest priority code of all ones is unassigned and denotes that no BIU has requested the bus. Each BIU gates his priority code onto the bus one bit at a time, starting with the most significant bit, and then compares the value on the bus with the bit. If the bus matches the transmitted bit, the process is repeated for the next most significant bit. If a mismatch occurs, then that BIU has lost the use of the bus (since some other BIU has put a higher priority code on the bus) and does not gate the remaining priority code bits of lower significance onto the bus. The BIU which survives all six bits and matches the entire bus gains allocation of the bus. The priority scheme takes advantage of the property that logic zeros will pull down logic ones on a multiparty open collector type bus.

Figure 2-7 illustrates a sample overlay priority poll between six users. In the first bit of the poll, user 30's one is pulled down by user's 13, 08, 04, 03, and 01 zero, and user 30 drops off. The process is repeated bit-for-bit until user 01 wins the bus. The jagged line indicates which priority users are participating in the bus allocation, by gating bits onto the bus, for each bit of the priority code.

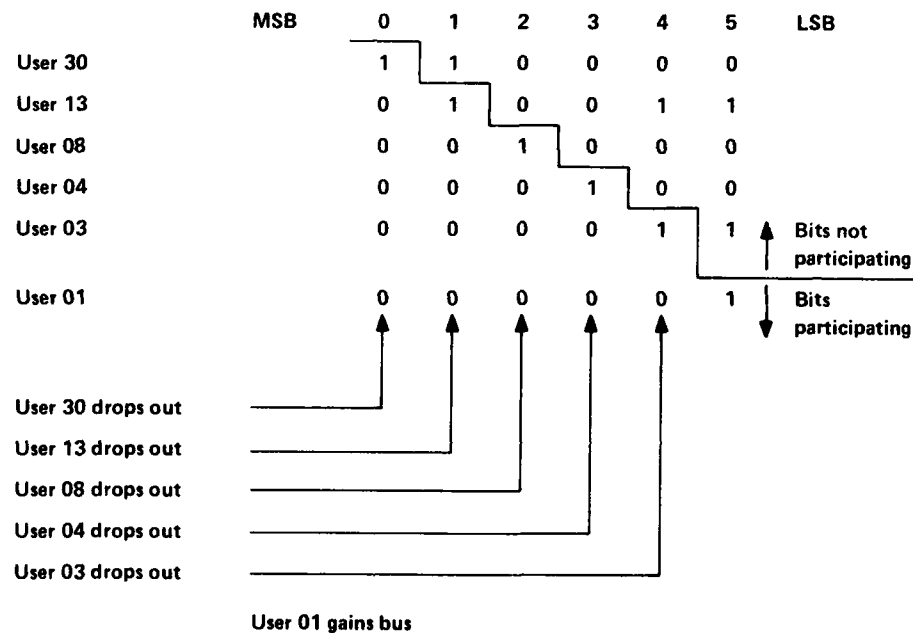


Figure 2-7. Sample Overlay Priority Poll

In this manner, 63-way priority is resolved in just one, distributed, bus sequence. Units not requesting the bus do not enable their drivers, consequently, the bus unassigned code is all ones. The BIU that used the bus last must continue to transmit the priority tag and raise Transfer Out until the six least significant bits of Bus Out indicate that the bus has been assigned.

Sample Write Message Protocol: (See Figure 2-6). Following allocation, the BIU gaining the bus places the sink address in the low six bits of Bus Out, encodes the address tag and raises Transfer Out. The addressed device responds and acknowledges selection by echoing the address on Bus In, encoding the address tag and raising Transfer In. Since the messages are DC interlocked, the source device now drops Transfer Out to acknowledge the correct echo address on Bus In. The sink device responds by dropping Transfer In.

The process is now repeated with the source device encoding the command on Bus Out and the sink device indicating its ability to execute the command (in this case a write) by encoding status on Bus In. If the device is unable to accept the command the sequence ends. If the device status is acceptable, the data block is transferred using both edges of Transfer Out and Transfer In. The sequence is completed when the sink device encodes final status on Bus In. Final status indicates if any errors were incurred during the data transfer sequence.

BIU Complexity/Cost: The parallel interface, absence of Manchester encoding, straightforward allocation method and DC interlocked interface handshake make this the simplest BIU of the bus configurations discussed. Since the last BIU that used the bus is responsible for re-transmitting the priority tag until the bus is allocated, the monitor function can be implemented as a "no message" timeout. This results in a simple monitor implementation.

2.4.3.5 Distributed, Tag Encoded, Serial, Selector Channel

Bus Description: Distributed control, serial, Manchester-encoded data bus. Serial version of bus described in the previous section. Last device to use the bus is responsible for the next allocation. Allocation takes N bus cycles to resolve 2^N requests (e.g., six cycles for 64 users). Allocation does not result in data collisions.

Number of Lines: One line.

Signaling Rate: 2 MHz

Data Rate: 0.8 Mb/s (without overhead).

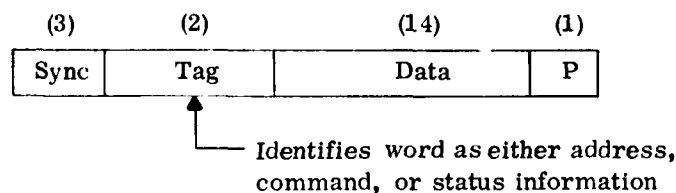
Length: 100 to 300 feet.

Terminal Attachment: AC coupled via coupling transformers.

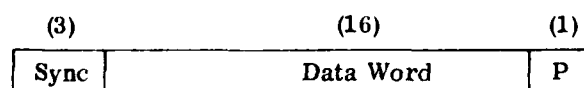
Error Checking: Bus parity, invalid command. Both command and data retry. System monitor required to check and recover bus allocation failures.

Message Formats:

20-bit command/adr/status/word



20-bit data word



20-bit priority poll command

(3)	(2)	(14)	(1)
Sync	Priority Tag	Priority Mask	P

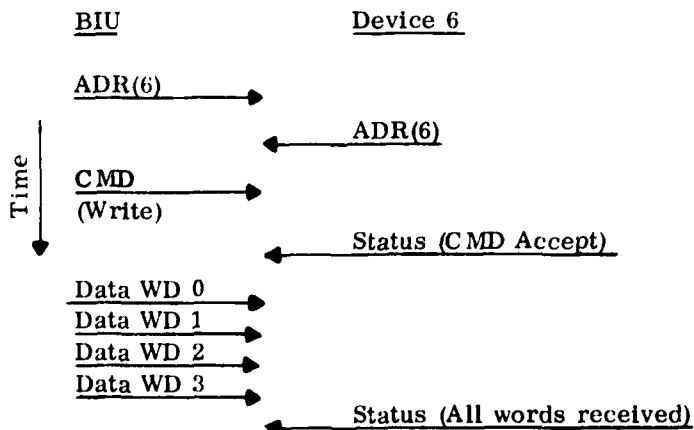
Two Manchester sync patterns are used. One pattern is for both the command/address/status word and the short priority poll message. The remaining sync pattern uniquely identifies a data word.

Access Method: The serial version of the distributed selector channel uses the same overlay, binary, priority scheme as the parallel channel described previously. In this case the bit-for-bit priority resolution is adapted to a serial Manchester interface. To accommodate the Manchester channel, the code of all ones has highest priority, and logic Manchester ones override, and have priority over, logic zeros.

The last BIU to use the interface transmits the six-bit priority poll message. Each BIU monitors the bus and decodes the priority poll message. Upon decoding the priority poll message each BIU transmits a Manchester one if the MSB of its priority number is a one, and does not transmit anything if its MSB is a zero. Units having zeros monitor the bus for ones. If a zero bit encounters a one it has lost this bit of the poll and does not participate in the remaining bits of less significance. The priority resolution continues in this manner, bit-for-bit, with the last BIU to use the bus transmitting N priority poll messages to resolve 2^N way priority.

For example: 64-way priority can be resolved in six priority poll frames. Each of the six poll frames is defined by the transmission of the priority poll message and a gap time during which BIUs which have logic ones in their priority code, for the bit position under contention, transmit Manchester ones on the bus.

Sample Write Message Protocol: The protocol is identical to the parallel selector channel with the exception that the address, command, status and data exchanges are encoded into 20-bit, Manchester, serial messages. A typical write sequence to device 6 of four data words follows:



BIU Cost/Complexity: The Manchester encoding and parallel-to-serial registers result in a moderate to expensive BIU. Since the priority poll message must be retransmitted until the bus is allocated, the monitor function can be implemented as a timer. This results in a relatively straightforward monitor implementation.

2.4.3.6 MIL-STD-1553B Data Bus With Overlay, Binary, Bit-for-Bit, Priority

Bus Description: Distributed control, Manchester encoded, serial bus. Last BIU which uses the bus is responsible for the next bus allocation. Bus allocation takes N bus cycles for 2^N priority. Bus access method is identical to the method used on the tag encoded, serial, selector channel described in subparagraph 2.4.3.5. Allocation does not result in data collisions.

Number of Lines: One line.

Signaling Rate: 2 MHz.

Data Rate: 0.8 Mb/s (without overhead).

Length: 100 to 300 feet.

Terminal Attachment: AC coupled via coupling transformers.

Error Checking: Parity, invalid Manchester, invalid command, message retry. In addition, a monitor function is required to monitor and recover allocation mechanism in case of a failure. Monitor can be implemented as a timer.

Message Formats: In addition to the standard 20-bit command, data and status words a new Manchester sync code should be developed for the priority poll. This new Manchester sync would serve the same purpose as the priority poll tag used in the serial, selector channel. An alternative approach to a new Manchester sync is the assignment of a 20-bit broadcast command for the priority poll message.

Bus Access Method: Same as the access method described in subparagraph 2.4.3.5 for the serial, distributed selector channel.

Sample Write Message Protocol: Same as stationary master MIL-STD-1553B channel (once bus control has been established by the access method).

BIU Cost/Complexity: BIU cost is more expensive than a standard 1553B channel, due to the access method and monitor function hardware.

Monitor can be implemented as a timer. Overall hardware cost is moderate to expensive.

2.4.3.7 Overlapped Control, Parallel, High Performance, Distributed Bus

Bus Description: This bus was developed for signal processing applications which require high data throughput and high bus efficiency. It consists of the following three sub-buses operating in parallel:

1. Parallel data bus
2. Serial allocation bus
3. Serial command/status bus

The serial buses are used to control the allocation of, and pass commands and status associated with, the parallel data bus. Allocation uses the overlay, binary bit-for-bit method described earlier and does not result in collisions.

Number of Lines: 22 lines total
17 lines parallel data bus
1 line serial allocation bus
1 line serial command bus
3 control lines - 2 central, 1 distributed

Signaling Rate: 8 MHz - data bus, 2 MHz - serial buses

Data Rate: 120 Mb/s (bus has no overhead).

Length: 60 feet (max). Longer length possible with degraded data rates.

Terminal Attachment: DC coupled with open collector or pulled up tri-state logic drivers.

Error Checking: Parity check on all three buses. Retry on error for data and command/status bus. Two centralized timing signals are used: a frame sync and a control strobe. A monitor function must be included which will provide an alternate timing source if the centralized timing signals fail. Also, if one driver fails the bus allocation serial sub-bus to a logic zero, automatic monitoring and switchover to an alternate allocation sub-bus must be implemented.

Message Formats: All data transfers are block transfers of 256 sixteen-bit data words. Each block of 256 data words is referred to as a bus frame. Following each 256-word data block, a 2 μ s wait interval is observed which allows for driver switchover. Each frame cycle is divided into 256 data strobe clocks and 68 bus allocation sub-bus and control sub-bus clocks. One 2-MHz allocation/control bus clock occurs for every four data bus clocks. Therefore, each bus frame cycle can be thought of as being divided into: 256 data transfer intervals, 68 bus allocation intervals, and 68 command bus intervals. This relationship is illustrated in Figure 2-8.

In one frame cycle the three data bus intervals are subdivided into the following major functions:

1. A data block is transferred during all 256 intervals of the data sub-bus. The data bus user is determined by the contention poll performed in the previous frame cycle.
2. The "ready" or "not ready to receive" state of all users is broadcast on the bus allocation and command sub-bus in sync with the centralized command strobe. The 63 buffer ready bits occupy bus allocation and command sub-bus intervals 0 to 31.
3. Contention for the use of the data bus in the next frame cycle is resolved during bus allocation sub-bus intervals 34 to 43.
4. Contention for the use of the command sub-bus in the current frame cycle is independently resolved during command sub-bus intervals 34 to 43.

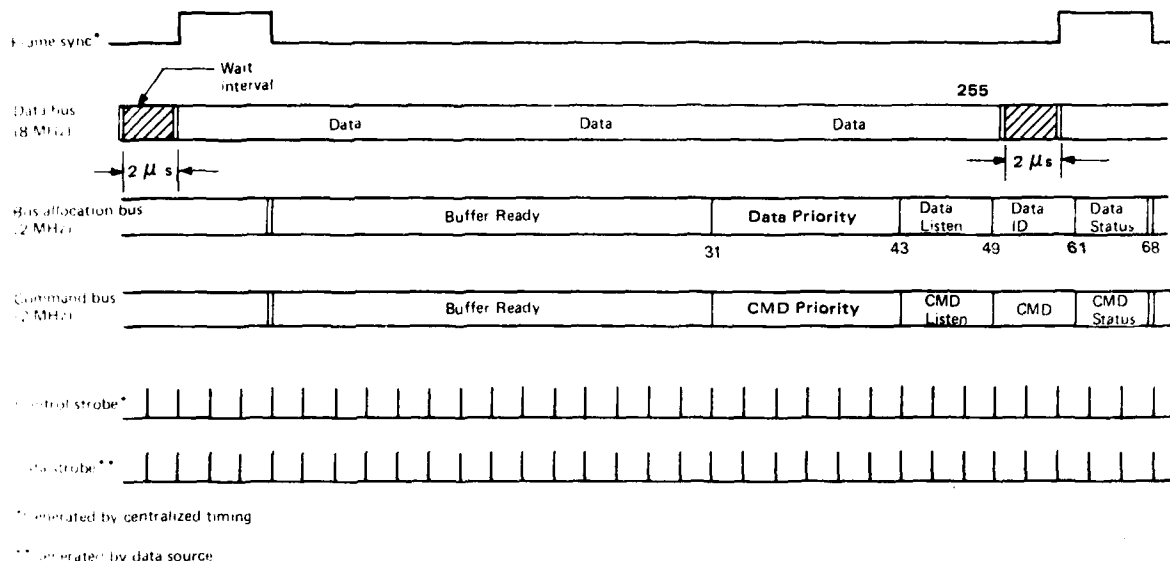


Figure 2-8. Overlapped-Control, High-Performance, Distributed Bus

5. Listener addresses for the next data frame and the current command frame are identified on the bus allocation sub-bus and the command sub-bus intervals 44 to 49, respectively.
6. The user which gained the command sub-bus transmits a command during sub-bus intervals 50 to 61.
7. The listener associated with the command transfer returns status during command sub-bus intervals 64 to 67. The listener associated with the current data frame returns data status during bus allocation bus-bus intervals 64 to 67.
8. The user which gained the use of the data bus-bus for the next cycle transmits a code identifying the type of data block to be transferred during bus allocation sub-bus intervals 50 to 57.

Access Method: A potential bus user first determines if his listener address is ready to receive data by monitoring the appropriate buffer ready bit broadcast by the listener during bus allocation or command sub-bus intervals 0 to 31. If the listener is ready to receive data, the user then proceeds to contend for the data bus. In this manner the contention mechanism and the data bus are not tied up with data transfers to busy listeners, that must be rescheduled and then repeated.

Bus contention is the serial bit-for-bit, overlay, binary contention scheme described in subparagraph 2.4.3.4. Logic zero codes override logic ones and all users gate their code onto the bus but at a time starting

with the MSB. The same contention scheme is independently duplicated for both the next data frame and the current command frame.

Sample Write Message Protocol: To write a data frame a user observes the following sequence:

1. Determine if the listener address is ready to receive
2. Contend for and gain use of the data bus in the next frame cycle
3. Transmit the listener address and the data ID. In this case the data ID identifies the block as write data
4. Send the block during the next frame cycle
5. Receive listener data status at the end of the next frame cycle

BIU Cost/Complexity: Most complex BIU described in this section. However, the additional hardware cost is offset by the high bandwidth and the absence of any command and control overhead. Duplicate centralized frame sync and control strobe monitors and generators must be included in any fault tolerant system design.

2.4.3.8 Overlapped Control, Serial, Distributed Bus

Bus Description: Serial version of the overlapped control, parallel data bus described in the previous section. In this implementation the data sub-bus is serial and is divided into 68 intervals as are the bus allocation and the command sub-buses. Since the data bus is serial, the data block size is 64 bits or four 16-bit words. Bus allocation and command sub-bus formats and access methods are identical to the parallel bus. All data buses are nonreturn-to-zero (NRZ) encoded.

Number of Lines: 6 lines total

- 1 line serial data bus
- 1 line serial allocation bus
- 1 line serial command bus
- 2 central control lines
- 1 distributed control line

Signaling Rate: 1.0 MHz

Data Rate: 1.0 Mb/s (bus has no overhead)

Length: 120 feet (max). (Longer length possible with degraded rates.)

Terminal Attachment: Same as parallel bus

Error Checking: Same as parallel bus

Message Formats: Bus allocation and command sub-bus formats and access methods are identical to the parallel bus. The data sub-bus is serial and is divided into the same number of intervals as the other sub-buses, that is: 68. The source of the data also provides the data strobe signal.

Access Method and Sample Write Message Protocol: Same as parallel bus.

BIU Cost/Complexity: Overall BIU would be complex. Again, the lack of command/control overhead may justify the additional hardware cost.

2.4.4 CANDIDATE BUS SELECTION

Paragraph 2.4.2 described in detail the criteria, requirements, and concerns for selecting a distributed data bus for a fault-tolerant processor network. In paragraph 2.4.3 eight data buses were described. In this paragraph the eight data buses are evaluated against all of the criteria described in paragraph 2.4.2. Of the eight buses, three buses were to be selected for further study.

Table 2-27 is an evaluation matrix in which the eight data buses are rated against the criteria. Each data bus was scored on a scale of 1 to 10 against each of the criteria. A perfect score of 10 indicates that the criteria was completely satisfied. Table 2-27 also contains the subtotals for each major area (system organization, bus access method, message structure and electrical/technology), and the total score for each data bus.

Figure 2-9 is a bar graph illustrating the relative merit of the eight data buses for system organization, bus access method, message structure and electrical technology. Figure 2-10 is a bar graph of the eight overall bus scores.

The buses having the highest scores show a strong combination of system organization and message structure. However, only one bus, which had a low electrical/technology score, had a high overall score; that is the parallel overlap bus. In the case of the overlap bus, the low electrical/technology score was offset by strong system organization and access method.

Obviously, the best buses show a good mix of all selection criteria. The buses having the three highest scores are:

1. Serial selector bus
2. Parallel selector bus
3. Overlap serial bus

However, the fourth ranked bus, MIL-STD-1553B with overlay priority, differed from the third ranked bus by only a few points.

Of the four highest rated buses three of the buses have these common characteristics:

1. Overlay priority
2. Serial (Manchester or NRZ)
3. AC coupling.

It would appear that these characteristics should be considered in any future trade-off studies.

Since there is no clear distinction between the third and fourth ranked bus, it is recommended that the top four buses be evaluated. The quantitative analysis in the Phase II study should crisply separate the differences between these buses.

TABLE 2-27. DATA BUS SELECTION MATRIX

Selection Criteria	Stationary Master MIL-STD-1553B	Non-Stationary Master	Contention	Distributed, Tag Encoded Selector		MIL-STD-1553B With Overlay Priority	Overlapped Control High Performance	
				Byte Parallel	Serial Manchester		Parallel	NRZ Serial
<u>System Organization</u>								
1. Decentralized Control	0	3	4	10	9	9	9	9
2. Error checking	7	7	6	9	9	6	7	7
3. Efficient protocol	1	2	3	8	6	7	10	10
4. Closed-loop control	3	3	3	10	10	3	5	5
5. Busy holdoff	0	0	0	10	10	0	8	8
6. Data throughput	4	4	4	7	4	5	10	6
7. BIU cost/complexity	5	1	1	7	3	3	0	1
	20	20	21	61	51	33	49	46
<u>Bus Access Method</u>								
1. Straight-forward allocation	0	1	0	9	8	8	8	8
2. Easily monitorable	0	1	1	7	6	6	6	6
		2	1	16	14	14	14	14
<u>Message Structure</u>								
1. Expandable CMD/status	3	3	5	8	8	3	7	7
2. Expandable address	3	3	5	8	8	3	7	7
3. Broadcast mode	10	10	10	10	10	10	10	10
4. Associative addressing	0	0	10	10	10	0	10	10
5. Block transfers	6	6	10	10	10	6	8	8
	22	22	40	46	46	22	42	42
<u>Electrical and Technology</u>								
1. High noise immunity	8	10	8	3	8	8	3	5
2. Minimum interface lines	10	10	10	3	10	10	3	6
3. Low signaling rate	9	9	9	9	9	9	2	9
4. Independent from gnd shift	10	10	10	2	10	10	2	2
5. Clock to data skew	10	10	10	2	10	10	2	5
6. AC coupled	10	10	10	0	10	10	0	0
	57	57	57	19	57	57	12	27
TOTALS (maximum 200)	99	101	119	142	168	126	117	129

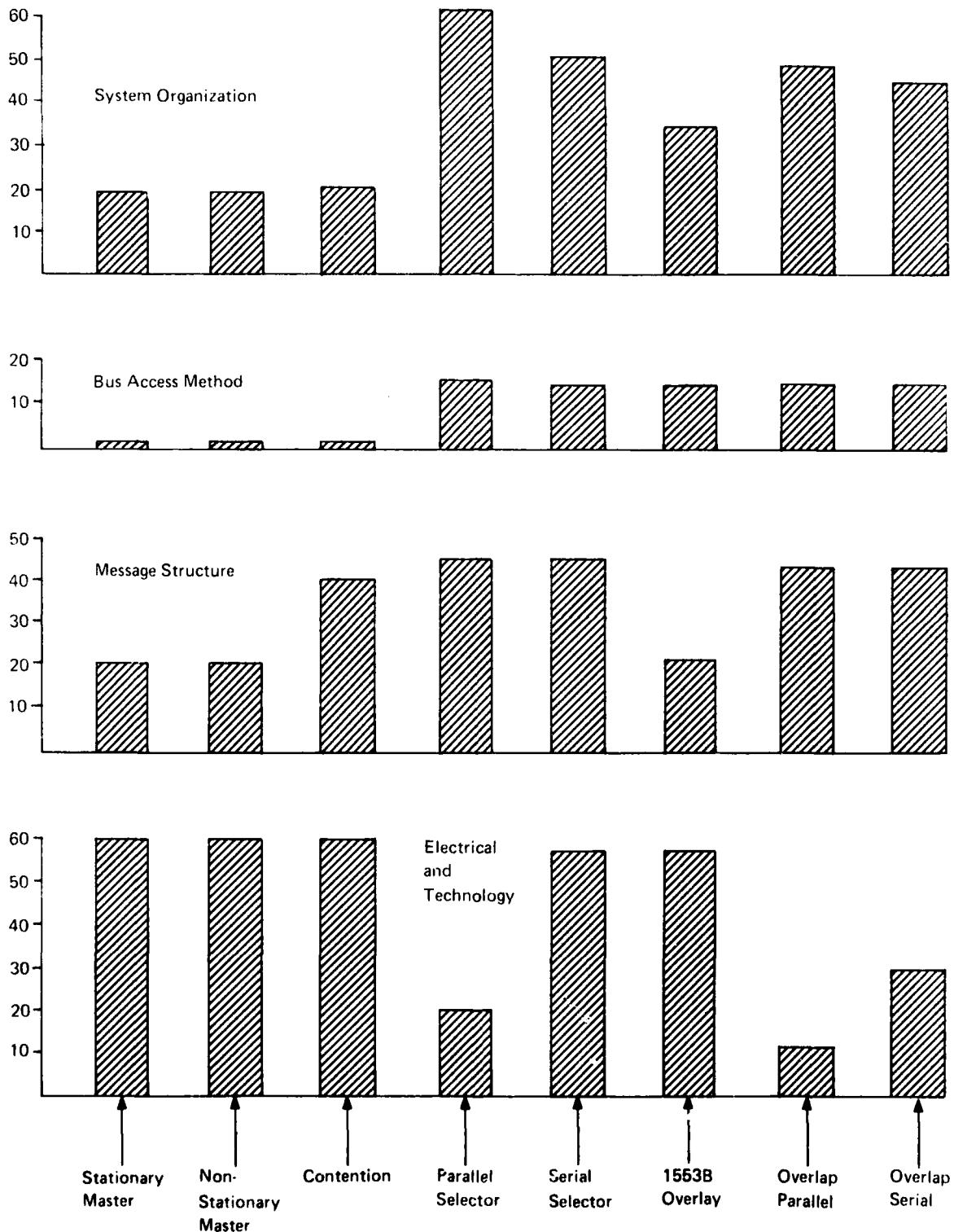


Figure 2-9. Bus Comparison Breakdown

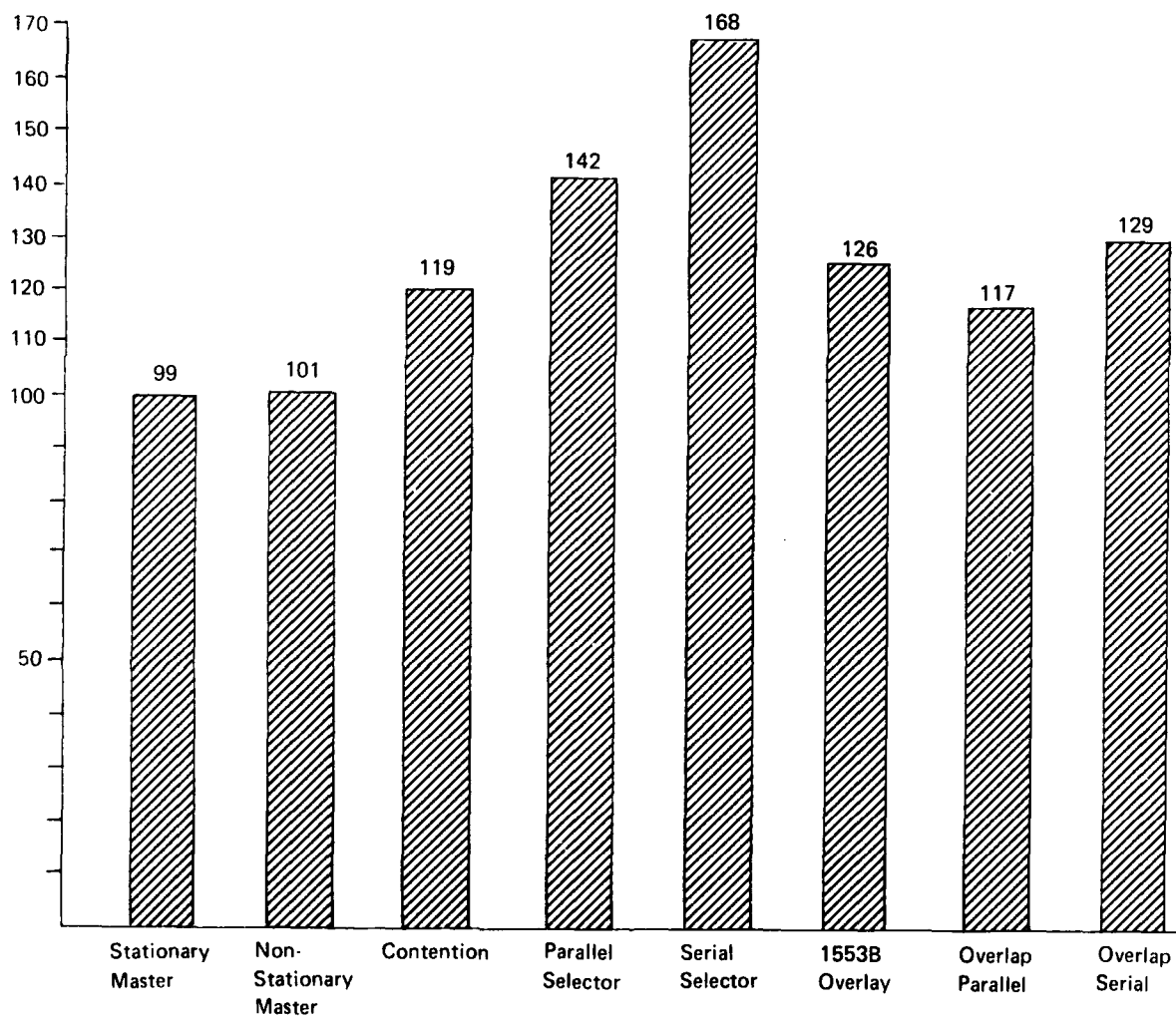


Figure 2-10. Overall Bus Comparison

2.5 TASK 1,6 - SURVEY OF MODELING TECHNIQUES

As indicated previously, the Phase II study will require modeling capabilities in three separate areas; namely, performance modeling, reliability modeling, and life-cycle cost modeling. The following sections discuss each of these areas, and describe the approach recommended for Phase II.

2.5.1 PERFORMANCE MODELING

The Federal Systems Division of IBM has regularly used performance modeling to aid in the design and development of complex systems. It is used to assist design trade-off studies during the concept development stages of a project as well as for design tracking during development of a system.

Simulation can generically refer to a number of specialized products or activities that can be part of the system development process: simulation, simulators, stimulators, emulators, and modeling. Each of these are distinctly different and should be defined.

1. A model is an abstract characterization of a system. It may be a physical entity or a program which represents the design of the system. A computer system model is usually a program designed to determine system utilization, operator responsiveness, resource contention, design sensitivities or bottlenecks. Computer system models are also known as performance models.
2. Simulation is the imitated execution of the system using the model. It is the process of using the model to do analysis.
3. A simulator may be a piece of hardware or a program designed to reflect the functional capabilities of the item being simulated. It provides a surrogate function for an absent, non-operational or undeveloped item. It will receive data over the operational interface and provide the proper responses to it, but it will not necessarily operate on the data with the real algorithms.
4. An emulator also reflects the functional capabilities of an item, but in addition, it also functionally behaves like it at the machine cycle level. For example, a processor emulator is driven by the actual programs of a system and simulates their execution on a microcycle basis. It simulates the execution of the actual piece of equipment, including proper timing relationships and statistics for each of the microcycles, to the code that will actually run it.

5. A stimulator imitates the external environment to a system under development or test. It may be a piece of equipment or a program that generates sequences of events or data for the testing of the operational system. It may be as simple as a test driver or may be sufficiently complex to simulate timing fidelity to real-time systems.

FSD addresses each of these types of simulation in five ways: processor emulation, interface simulator and stimulator, environment simulator and stimulator, computer system performance simulation and modeling, and application system simulation and modeling. The latter two are both abstract characterizations of a system but have totally different emphasis and products. The models developed for each of these are different in function, implementation and mathematics:

1. Computer system simulation focuses on the computer components of a system with all of its processing, interactions and external scenarios. Its purpose is to verify the configuration and design and to aid the design and development process.
2. Application simulation is an abstract characterization of the total physical system, of which computers are only a part. It would include the simulation of all the satellites of a navigational system, all the aircraft and ships of a tactical exercise, all the railroad cars of a railroad yard, etc.

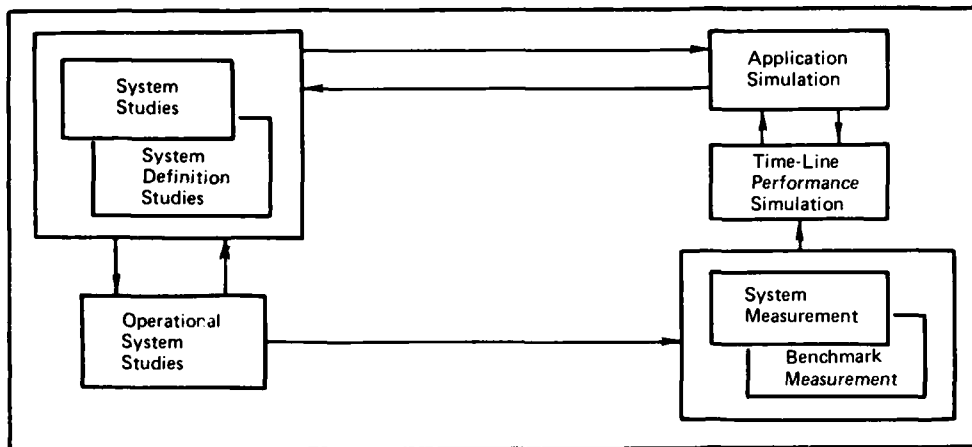
Figure 2-11 depicts the relationships or dependencies that may exist between the various types of simulation for each phase of a system's development process. Computer system performance simulation has been presented at two different levels of detail in this figure: system time-line simulation and software prototype performance simulation. Each will be defined in Section 2.5.1.1.

Model calibration is the technique of validating the estimates used in a model. It usually involves measurement of a developing system as early in the development process as possible. Measurement activities are depicted accordingly in Figure 2-11.

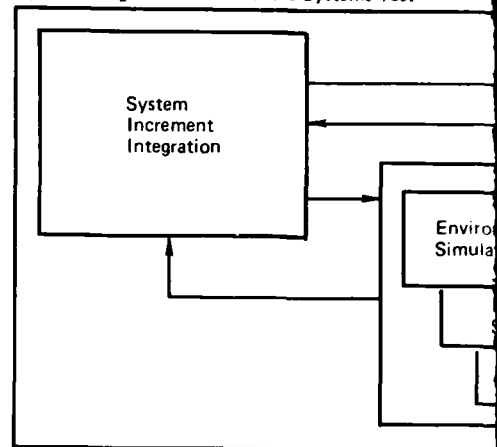
Table 2-28 identifies the specific activities of the system development process which should use the different types of simulation or modeling.

2.5.1.1 Survey of Performance Modeling Techniques For The Fault Tolerance Study

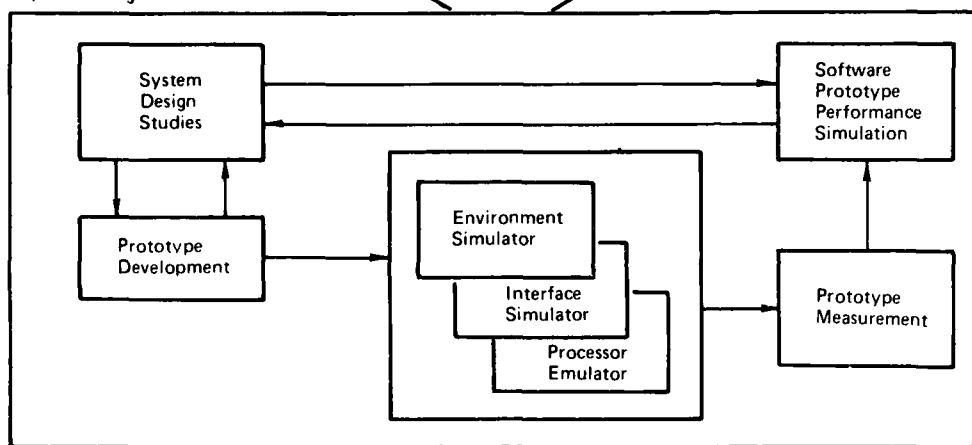
Four approaches to computer system performance modeling exist: paper-pencil, analytic, hybrid (or system) time-line, and discrete software prototype models. They vary in level of detail, estimate granularity, fidelity, and cost. The choice of approach depends upon the project characteristics, requirements for accuracy, budget, and timeliness.



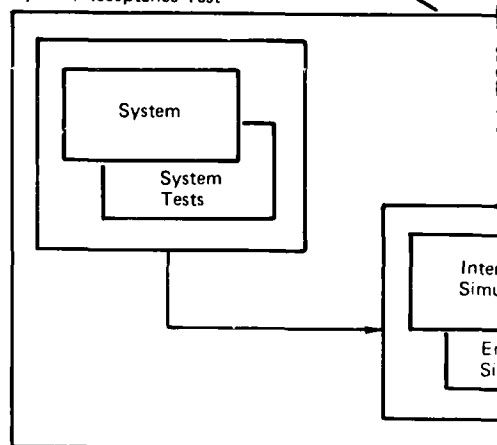
System Integration and Software Systems Test



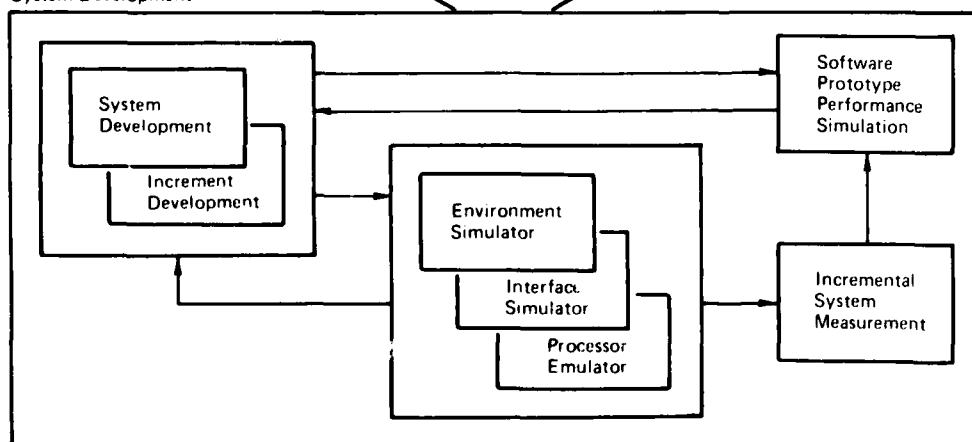
System Design



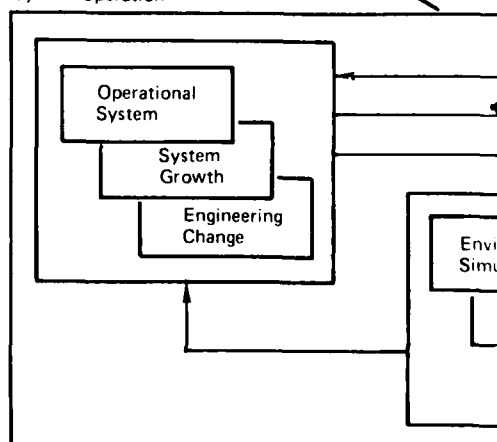
System/Acceptance Test



System Development

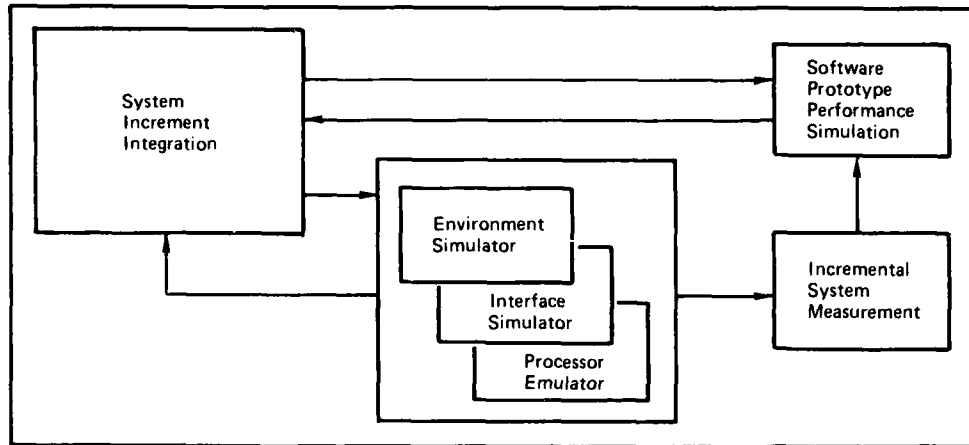


System Operation

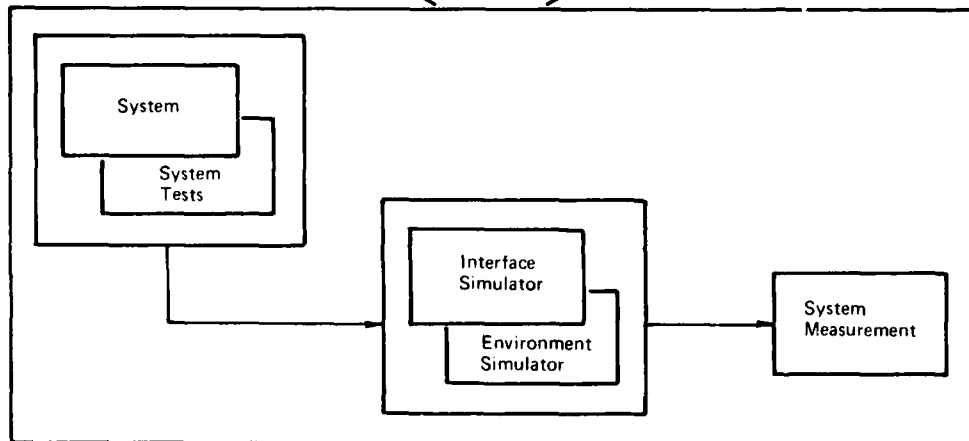


The Relationship Between the System Simulation & Measurement Practices to the System Development Process

System Integration and Software Systems Test



System/Acceptance Test



System Operation

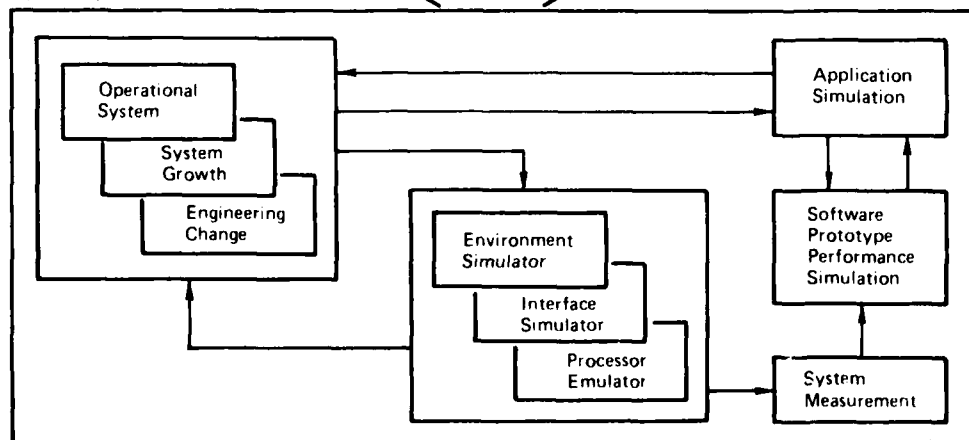


Figure 2-11. Simulation Relationships

TABLE 2-28. PRIMARY ROLES OF SOFTWARE SIMULATION

Software Process Activities	Software Simulation Types				
	Application Simulation	Computer Performance Simulation	Environment Simulator	Interface Simulator	Processor Emulator
System Definition	Concept Formulation Analysis	Requirements Allocation Analysis			
Software Design		Design Trade-Off Analysis	Prototype Development	Prototype Development	Prototype Development
Software Development	Development Change Analysis Support	Design Control Analysis	Subsystem Test Support	Subsystem Test Support	Unit Test Support
Software System Test	Test & Integration Support	System Tuning	Test & Integration Support	Test & Integration Support	
System/Acceptance Test			Acceptance Test Support	Acceptance Test Support	
Operations and Maintenance	Design Change Analysis	Design Change Analysis	Training Maintenance Support	Training Maintenance Support	Maintenance Support

The four approaches are listed in ascending order according to accuracy, fidelity and cost as follows:

1. A paper-pencil model is a mathematical representation of a system whose analytic calculations do not have to be performed iteratively. It is sufficient for approximating resource use for systems that do not have asynchronous processes and for which stress situations are easily characterized.
2. An analytic model is a software implementation of the mathematical model of the computer system. This technique is useful for analyzing systems that have numerous periodic processes which may have asynchronous execution. Analytic models can be used for estimating resource loading for a time-line or for network modeling. They can be used for analysis where instantaneous resource contention or bottleneck identification are not requirements.
3. A hybrid time-line is a software model that provides a discrete simulation of sequences of events within a time-line but uses analytic techniques for other functions of the model. A sequence of events may be deterministic or probabilistic. The order of processing and I/O is uniformly distributed within a function. I/O and data base activity is approximated without concern for location or sequence. However, each hardware resource and

top-level software component is represented. Such models are required for analysis of systems have a periodic activity.

4. A discrete software prototype model is a more detailed and totally discrete version of the hybrid time-line model. It contains service level models for the key operating system services, discrete estimates for inter-service application execution, and discrete execution of each input/output action. Such models are excellent for identifying and diagnosing discrete resource contention problems.

The characteristics of the four models are summarized in Table 2-29.

2.5.1.2 Fault Tolerance Performance Model Selection

The Phase II fault tolerance study is to be a configuration and design trade-off study. The tradeoffs will be performed with the F-15 as the prototype system. This means that since it is an existing system sufficient a priori information exists for relatively detailed design trade-offs and that measurement data can be collected to support them if necessary. The objectives of these tradeoffs will be to determine whether a given fault-tolerant architecture (system configuration and function allocation) can satisfy the performance requirements of the target system.

2.5.1.2.1 Fault Tolerance Study Model Requirements

Given that the Phase II study is going to be a comprehensive and exhaustive study of numerous future fault-tolerant system designs and that evaluation of the fault-tolerant designs are to be based on the F-15 functional requirements, this allows the deduction of the following requirements for the performance model:

1. It should be easy to redefine configurations and functional allocations within the distributed avionics environment.
2. It should be easy to redefine operational scenarios.
3. The model should provide response time distribution statistics.
4. The scenarios should be deterministic and thus repeatable.
5. The model should allow for the detection of system resource contention.
6. The model should provide a comprehensive set of statistics.
7. The model should be relatively inexpensive to develop and use.
8. The model should be easy to learn and easy to use.

2.5.1.2.2 Performance Model Proposed for the Fault Tolerance Study

Although the Federal Systems Division of IBM regularly applies each of the four modeling approaches, the requirements of Section 2.5.1.2.1 led us to select the hybrid time-line approach for the fault-tolerance study. The hybrid time-line capabilities satisfied all of the above requirements, whereas the other approaches did not. It is obvious that a

TABLE 2-29. PERFORMANCE MODELING APPROACH CHARACTERISTICS

Approach	Analysis Capabilities	Liabilities	Language
Paper-Pencil Models	<ul style="list-style-type: none"> o Applicable to systems having limited number of well defined analysis points o Applicable when an "order of magnitude" estimate is sufficient 	<ul style="list-style-type: none"> o Detailed time-line characterization is laborious and therefore inefficient and error prone o Consideration of random, asynchronous events is difficult o Detection of contention problems is difficult and laborious o Analysis of multiple interdependent variables is laborious o Response time calculations for more than a few simple cases is laborious 	<ul style="list-style-type: none"> o Not applicable
Analytic Models	<ul style="list-style-type: none"> o Can handle simple or complex problems, depending on the number and order of the equations used to represent the system o Can be used for systems which have asynchronous, steady-state behavior o Sophisticated analytic models can estimate resource queueing o Can estimate average response times 	<ul style="list-style-type: none"> o Require the data describing the system to be reducible to a set of equations that can be solved directly or by iterative techniques o Require different sets of equations to be developed for each configuration or set of functional allocations o Require significant a priori knowledge of the performance characterizations of the system interrelationships o Are inefficient for studying detailed system data flow 	<ul style="list-style-type: none"> o Any higher level scientific language
Hybrid Time-Line Models	<ul style="list-style-type: none"> o Provide discrete time-line simulation of system workload o Are adaptable to changing scenarios o Can estimate resource queueing o Can estimate response time distributions o Can represent asynchronous, nonsteady-state processing o Can be generic o Are extendable in function and detail 	<ul style="list-style-type: none"> o Can't detect instantaneous contention or bottlenecks o Macroscopic view of the system 	<ul style="list-style-type: none"> o Any high-level scientific or simulation language
Discrete Software Prototype Models	<ul style="list-style-type: none"> o Provide detailed analysis of system design o Can detect and quantify instantaneous system bottlenecks and system contention points o Can provide discrete response time distribution statistics are adaptable to changing scenarios and configurations o Provide maximum fidelity with the simulated system o Are extendable in function and detail 	<ul style="list-style-type: none"> o The model can be quite complex o Development requires simulation skills o The largest of the performance modeling efforts, by a factor of two or three o The most costly to execute, possibly five to ten times more expensive than the other performance models o Are application or system unique 	<ul style="list-style-type: none"> o Any simulation language, computer system simulation (CSS), general-purpose simulation system (GPSS), SIMSCRIPT, extended computer system simulation (ECSS), etc.

paper-pencil model could not satisfy a number of the above requirements. Analytic techniques could not satisfy the first three requirements and the discrete software prototype level of modeling could not satisfy the last two.

An existing generic time-line model is proposed, PROTIME II. PROTIME II is a hybrid model that has evolved over the past 2 years to address function partitioning problems of distributed systems. It is a model that uses analytic techniques for resource use calculations and discrete event techniques for resource contention and event triggering and termination.

Very generally, PROTIME II's significant features are listed below. Detailed discussions of them are presented in the following section of this paper:

1. PROTIME II provides a simple simulation oriented terminal interface to the user. It uses the standard SPF/TSO facilities of System/370. It uses a preprocessor to translate the simulation parameters into an input data set for execution by a FORTRAN program.
2. The user interface requires no knowledge of simulation languages.
3. The user interface is table oriented, which facilitates the re-definition of operational scenarios, configurations, or function allocations. It also has an on-line tutorial capability that reduces the likelihood of errors, and reduces implementation costs.
4. A PROTIME II model can accommodate any level of detail.
5. PROTIME II uses analytic techniques for processor and device use and consumption statistics. It discretely simulates device and processor contention and function activation. This combination reduces the complexity of the user interface and reduces execution costs.
6. General output statistics are provided with more detailed snapshot statistics available as an optional output.

Note that some of the capabilities of this model will probably not be needed for the Phase II study, such as the device latency times.

2.5.1.3 Fault Tolerance Model

2.5.1.3.1 Background

The PROTIME II model was developed to provide a modeling capability with analytic ease of calculation, discrete changes in resource use requirements and input capabilities that would not require model building experience.

The PROTIME II model chosen for the fault-tolerance study is based on a precursor model named PROTIME (processor timeline model) that was originally developed for function allocation studies for multiple processor systems. It combined discrete simulation for resource contention with analytical calculation of the use of these resources. The

input and model parameters were designed to correspond to simple generic hardware and software element characteristics. The PROTIME model was written in BASIC for the IBM 5100 computer.

In 1979 an IRAD task was performed for the Avionic Systems branch of Federal Systems Division to extend the capabilities of PROTIME to handle a larger number of hardware and software elements and to execute faster for small and medium size models. In addition, a preprocessor was developed to convert input data from table oriented parameters describing the hardware and software elements to the generic parameters for these elements that are used in the model. The model was renamed PROTIME II.

Both PROTIME II and its preprocessor are written in PARA FOR (a structured FORTRAN) for use on a System/370.

2.5.1.3.2 General Description

PROTIME II is a generic model that determines hardware component use and software component execution times based on a user oriented description of a system's hardware resources, software, and workload. It determines resource use of system hardware and execution time of system workload components using generic elements for processors (CPU), devices (I/O devices), functions (workload) and events (occurrences that change system state). The model user provides input that describes resource availability for the hardware components (processors and devices), resource requirements for the workload (functions), and interactions within the workload (events).

Figure 2-12 conceptually presents the processing functions of a system. It is similar to a PERT chart in that the processing is depicted between the nodes with processing state changes, occurring at the nodes. It shows a simple time line of a single processor system. External event 1 could represent the scheduled arrival of radar fix data which starts functions 1 and 2. Function 1 logs and error checks the data while function 2 performs preliminary data conversion. External event 2 is a request to generate position information using functions 3, 5, 6; however, those functions cannot start until the raw radar fix data is processed, thus requiring internal event 1 which marks the completion of function 2 and external event 2.

Events are used to establish a time line of changes in system state that reflect changes in the demand for system resources. At each change in system state analytical equations are solved to determine resource use for that state. These changes in system state are recorded along with the resources used. After completion of the model run the statistics associated with resource use and execution of functions is tabulated and printed.

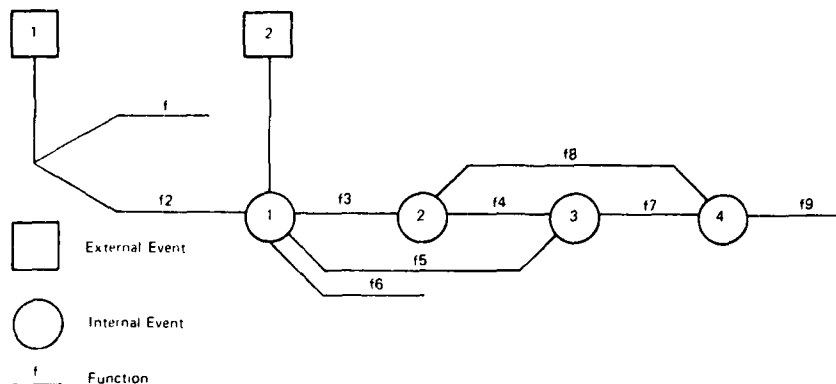


Figure 2-12. Sample Timeline - Single Processor System

2.5.1.3.3 PROTIME II Features

PROTIME II is composed of two components; the user interface which includes the input data preprocessor and the simulation execution model. Both these components are written in PARAFOR, a PL/I preprocessor that provides structured macros for use with the FORTRAN H extended compiler.

2.5.1.3.3.1 User Interface

The user interface to PROTIME II is designed to allow input data to be defined in terms applicable to the user. This data is entered into a set of skeleton tables via SPF/TSO. The data is converted, by a preprocessor, into the generic parameters required by the model. This approach was employed to maintain the generality of the model and allow the user to concentrate his effort on system analysis tasks instead of data input definition.

A typical SPF/TSO terminal session for executing PROTIME II consists of updating the input data set, setting up JCL pointers to the new data set and SUBMITting the job for batch execution. First a user would log on under TSO and select SPF. The update mode would be entered and the new input data set would be generated from an existing data set or a skeleton table using the standard SPF update functions.

When the new input data set has been updated and saved the JCL data set to execute the model is updated so that the DD statement defining the input data set points to the new input data set. When the JCL data set is updated a SUBMIT command is issued to submit the model as a batch job. The JCL data set is then saved and the SPF mode exited in the normal manner. Lastly, the user logs off TSO to end the session.

Instructions for updating each section of the input data set along with tips on using SPF/TSO are provided in a data set member named TUTOR. A sample of TUTOR is shown in Figure 2-13 and the skeleton input data set tables are shown in Figure 2-14 located in section 2.5.1.3.4.

PROTIME II: A102089		MEMBER: TUTOR		DATE: 01/01/05	
LIBRARY: PROMODEL		LEVEL: 01.22		TIME: 16:05	
TITLE: DATA		USERID: A102089		PAGE: 06 OF 20	

START	1	2	3	4	5	6	7	8	MOD PAGE	
									02710013	*
									02720013	*
									02730013	*
									02740013	*
									02750013	*
									02760013	*
									02770013	*
									02780013	*
									02790013	*
									02800013	*
									02810013	*
									02820013	*
									02830013	*
									02840013	*
									02850013	*
									02860013	*
									02870013	*
									02880013	*
									02890013	*
									02900013	*
									02910013	*
									02920013	*
									02930013	*
									02940013	*
									02950013	*
									02960013	*
									02970013	*
									02980013	*
									02990013	*
									03000013	*
									03010013	*
									03020013	*
									03030013	*
									03040013	*
									03050013	*
									03060013	*
									03070013	*
									03080013	*
									03090013	*
									03100013	*
									03110013	*
									03120013	*
									03130013	*
									03140013	*
									03150013	*
									03160013	*
									03170013	*
									03180013	*
									03190013	*
									03200013	*
									03210013	*
									03220013	*
									03230013	*
									03240013	*

FOR EACH PROCESSOR INCLUDED IN THE MODEL, THE USER MUST ENTER A ROW (I.E., LINE) OF DATA IN THE PROCESSOR SPECIFICATION TABLE. EACH SPECIFICATION INCLUDES:

NAME ANY ALPHA-NUMERIC LABEL (FROM 1 TO 8 CHARACTERS IN LENGTH): ANY NAME FOR PROCESSOR.

FIELD X.X (UP TO 6 DIGITS, OR 5 DIGITS PLUS A DECIMAL, IN LENGTH): PROCESSING SPEED, IN UNITS OF K-INSTRUCTIONS PER SECOND (I.E., 1000'S OF INSTRUCTIONS PER SECOND), WHERE $0 < N \leq 999999$. SINCE THIS IS A DECIMAL FIELD, THE RANGE OF POSITIVE VALUES IS FROM .00001 (OR 1 INSTRUCTION EVERY 100 SECONDS) TO 999999 (OR ALMOST A BILLION INSTRUCTIONS PER SECOND).

CORE (K) X.X (UP TO 6 DIGITS, OR 5 DIGITS PLUS A DECIMAL, IN LENGTH): SIZE OF PROCESSOR MEMORY, IN UNITS OF K-BYTES (WITH 1 K-BYTE EQUIVALENT TO 1000 BYTES OF CORE), WHERE $0 < N \leq 999999$ K-BYTES. (SEE DISCUSSION OF CORE IN SECTION 6.)

ACPU NOT AVBL N (UP TO 2 DIGITS IN LENGTH) OR BLANK: PERCENTAGE OF CPU OR PROCESSING WHICH IS NOT AVAILABLE TO USERS, IN UNITS OF PERCENTAGE POINTS, WHERE $0 \leq N \leq 99$. FOR EXAMPLE, 10 MEANS 10% OR TEN PERCENT. THIS FIELD CAN BE USED WHENEVER THE USER WISHES TO SIMULATE A PROCESSOR WHICH DOES NOT HAVE 100% PROCESSING POWER AVAILABLE FOR THOSE FUNCTIONS BEING SIMULATED. FOR EXAMPLE, THE USER MIGHT WISH TO ACCOUNT FOR SOME UNSPECIFIED BACKGROUND ACTIVITY. THIS FIELD CAN BE LEFT BLANK IF 100% CPU IS AVAILABLE. STATISTICS FOR THE PROCESSOR'S TOTAL UTILIZATION WILL INCLUDE ANY AMOUNT SPECIFIED HERE.

COMMENTS ANY ALPHA-NUMERIC DATA: USER COMMENTS

Figure 2-13. Sample Page of Tutor

2.5.1.3.3.2 Input Data Preprocessor

The preprocessor for PROTIME II was developed so that the model user could enter data in parameters applicable to his application. The preprocessor assigns entity numbers to the processors, devices, functions and events, and stores them with their associated data in indexed arrays for easy access by the execution model. The data describing each entity is entered in data units that are logical for that entity. The preprocessor converts these units to a consistent set of internal units for use by the execution model.

Linkages and interactions among the entities (processors, devices, functions and events) are described by the user in tables relating them by name. These names are converted to the proper indexes and stored in the required locations in the arrays representing the entities.

Figure 2-14. PROTIME II Model Data Entry Format

PROJECT: A302009
 LIBRARY: DPSMODEL
 TYPE: DATA

MEMBER: SKELETON
 LEVEL: 01.10
 USERID: W995477

DATE: 80/03/05
 TIME: 16:05
 PAGE: 02 OF 02

1	2	3	4	5	6	7	8
EVENT NAME	P E = I,X	I N G =A	E A T =R	N A P =#	TIME OF 1ST OCCURRENCE HH:MM:SS.TTT	REPEAT CYCLE HH:MM:SS.TTT	COMMENTS
XXXXXXXX	X	A	R	0	00:00:00.000	00:00:00.000	XXXXXXXX
							*03550005
							*00560005
							*00570005
							*00580005
							*00590005
							*00600005
							*00610005
							*00620005
							*00630002
							*00640000
							*00650000
							*00660000
							*00670000
							*00680000
							*00690010
							*00700009
							*00710009
							*00720009
							*00730009
							*00740009
							*00750009
							*00760009
							*00770009
							*00780009
							*00790002
							*00800002
							*00810000
							*00820000
							*00830000
							*00840000
							*00850000
							*00860000
							*00870000
							*00880000
							*00890000
							*00900006
							*00910006
							*00920000
							*00930000
							*00940000
							*00950000
							*00960000
							*00970000
							*00980000
							*00990000
							*01000000
							*01010000
							*01020000
							*01030000
							*01040002

SPECIFY FUNCTIONS

1	2	3	4	5	6	7	8
FUNCTION NAME	STARTING EVT NAME	ENDING EVT NAME	SYS PRIO	INIT & DONE	COMMENTS		
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXX	XX	XXXXXXXXXXXXXXXXXXXXXXXXXX		
							*00720009
							*00730009
							*00740009
							*00750009
							*00760009
							*00770009
							*00780009
							*00790002
							*00800002
							*00810000
							*00820000
							*00830000
							*00840000
							*00850000
							*00860000
							*00870000
							*00880000
							*00890000
							*00900006
							*00910006
							*00920000
							*00930000
							*00940000
							*00950000
							*00960000
							*00970000
							*00980000
							*00990000
							*01000000
							*01010000
							*01020000
							*01030000
							*01040002

SPECIFY FUNCTION REQUIREMENTS

1	2	3	4	5	6	7	8
REQUIREMENTS							
I/O ACTIVITY							
FUNCTION NAME	PROCESSOR NAME	CORE(K) (X.X)	KOPS (X.X)	DEVICE NAME OR TYPE ID	SIZE OF XFR (BYTES)	NUMBER OF XFRS	
XXXXXXXX	XXXXXXXX	XXXXXX	XXXXXXXX	XXXXXXXX	XXXXXX	XXXXXX	
							*00930000
							*00940000
							*00950000
							*00960000
							*00970000
							*00980000
							*00990000
							*01000000
							*01010000
							*01020000
							*01030000
							*01040002

Figure 2-14. PROTIME II Model Data Entry Format (Cont.)

During processing of the input data is checked for errors in format and consistency in processor, device, function, and event names. The input data is then written to a print data set. If no errors are found control is passed to the execution portion of PROTIME II.

The PROTIME II preprocessor is executed as a called subroutine of the main model so that it can be rewritten and compiled separately to provide specific capabilities for a given user application.

2.5.1.3.3.3 Execution Model

The execution portion of PROTIME II solves analytic equations to determine resource use at each change of system state as determined by the time line established from the system workload description.

Based on the workload description (functions and events) of the system a time line of system state changes is determined. A system state is defined as a set of processor and device resource availability and function resource requirements that produce a resultant system resource consumption. A system state change will occur whenever an event that causes a change in system resource consumption occurs. Each time a change of state occurs in the system analytic equations are used to calculate resource use based on the workload's demand for resources. The availability of system resources is then used to predict future events in the system workload which are used to dynamically alter the time line of system state changes.

Since the hardware and workload entities are basically generic and the analytical equations are solved in generic units (I/O transfer units, processor work unit, time units, I/O transfer units per time unit, etc.) any level of detail can be modeled by PROTIME II.

The processor entity can represent anything from a System/370 computer system to a disk or tape control unit; devices anything from a switch to a hyperchannel. Functions could be as general as "process all batch work" or as detailed as "signal transfer of one I/O word" and time unit can represent anything from nanoseconds to years. Thus, by appropriate selection of preprocessor conversions, any level of detail can be handled.

The execution speed of the model is not affected directly by the level of detail, only by the number of system state changes and the statistical output requested by the user. However, more detailed models generally result in more system state changes.

2.5.1.3.4 Simulation Parameters

Model input data is divided into six sections: run control information, processor description, device description, event description, function linkage specification, and function requirements description. The processor and device descriptions make up the system hardware specification and the event description, function linkage specification and function requirements description make up the workload specification. Figure 2-14 shows a skeleton of the data entry format and should be referred

to when reading the following sections. The discussions below apply to the PROTIME II preprocessor for use in the fault-tolerance study. The PROTIME II model simulation methodology is described in Appendix A.

2.5.1.3.4.1 Simulation Control Data

The simulation control information is composed of three items: title, date, and maximum simulation time. The title is a 53-character field that can contain any EBCDIC character and is used only to identify the run output listing to the user. The date, which is appended to the title during printout to aid in user identification of the output, is an eight character field which may contain any EBCDIC character. The maximum simulation time, which is used to terminate the simulation when the clock reaches the indicated value, is entered as three fields. The first two are two-digit fields representing hours and minutes, each followed by a colon (:). The third field is the seconds field which contains five digits, three to the right of the decimal point. This format (HH:MM:SS.XXX) is used to enter all time data.

2.5.1.3.4.2 Processor Descriptions

The processor description has four fields. The first is the processor name, composed of one to eight EBCDIC characters, which is used to identify the processor to the model and in the output listing. The second is the instruction processing rate in kiloinstructions per second (KIPS), which is entered as a floating-point number. Next is core size in kilobytes, which is also entered as a floating-point number. Last is the percent of the CPU that will not be available to the functions, which is entered as a two-digit number. This feature can be used to approximate the processing impact of a continuously executing resident function without modeling it. A column is also provided for optional comments.

2.5.1.3.4.3 Device Descriptions

The device description consists of four fields. The first field is the device name, which is used to identify the device to the model and in the output listing. It may consist of up to eight EBCDIC characters if only one device is specified. Instead of specifying a single device, the user may specify a group of devices with the same characteristics. To do this the user enters a one to six-character device type, left-justified, in the name field. The number of devices in the group is entered in the two-digit quantity field. (The quantity field must be blank for a single device.) The input preprocessor then generates a unique identifier (name) for each device by appending a two-digit number, from one to the quantity specified, to the six-character type designation. This identifier will be used in the model and the output listing. The user-entered description of device activity of the function requirements specification section must use this same identifier. The characteristics for all devices in the group are then specified by the entries in the remaining two device specification fields.

The next two fields are device characteristic fields: the device latency is specified in milliseconds and the device transfer rate in bytes/second. Both are entered as floating-point numbers. A comments column is also provided.

2.5.1.3.4.4 Event Description

Events may be one of two types; internal or external. Internal events occur as a result of function completion or other event occurrences; external events occur solely as a function of time. Both types of events are described in the same skeleton with four parameters used for internal events and five for external events.

The first two parameters of the event description, event name and type, are common to both internal and external events. The name may consist of from one to eight EBCDIC characters and is used to identify the event to the model and in the output listing. The type designation is a one-character field that must be I for internal or X for external. Internal events have two other fields. They are the "anding" flag and the repeat flag, both single-character logic flags. If an A (or any other nonblank character) is entered in the first field, the event is considered to be an "anding" event (all input events and functions are required to occur before the defined event occurs). If the field is blank, an "or" event (when any one of the possible event or function inputs occur the event will occur) is created. The repeat flag, if set to R (or any nonblank character), will cause the event to be retained (allowed to reoccur as often as its input requirements are satisfied) by the model after its occurrence. If the repeat flag is left blank the event will be deleted by the model following its occurrence.

Three more fields are used to describe external events. The first of these fields is a two-digit snapshot identifier for snapshot events. Snapshots are printouts of selected parameters during the execution of the model and will be described more fully in the output description section. If this field is left blank, a nonsnapshot or "normal" external event will be created. The next two fields are the first execution field, which specifies the time of the first occurrence of the event, and the event repeat cycle, both entered in the time format discussed in Sub-section 2.5.1.3.4.1. The first time of occurrence. If the repeat cycle is zero, the event will occur only once. A comments column is also provided for both types of events.

2.5.1.3.4.5 Function Linkage Specifications

The linkage between functions are specified by five parameters. The first parameter is the function name, which may be one to eight EBCDIC characters. The second parameter is the starting event name which is used to identify the event that will trigger this function. Next is the ending event name which identifies the event that is to occur when the function completes. All event names are one to eight EBCDIC characters. The function and event names must be the same as those used when specifying the functions and events.

Following the event names is the function system priority, entered as a four-digit number. The larger the number the lower the priority.

The final parameter is a two-digit number indicating the function's initial percent complete. If this field is zero, the function is not initially active in the model. If the field is nonblank, the function will be made active when the model is initialized, with the number entered being the percentage of its resource requirements (both processor and device) that have already been consumed. A comments field is also provided.

2.5.1.3.4.6 Function Requirements Description

Seven parameters must be provided to describe each function's resource requirements. The first parameter is the function name which may be up to eight EBCDIC characters. The next three parameters are associated with processor requirements. First is a one to eight EBCDIC character name which identifies the processor. Next is the function's core requirements in kilobytes followed by the instruction count in kiloperations per second (KOPS). These last two parameters are entered as floating-point numbers.

There are three parameters in the function requirements description that are used to define device requirements. First is a one to eight EBCDIC character device name or type and number identifier. This is followed by size of this function's data transfers to this device, and the number of such transfers per execution of the function. The I/O transfers are uniformly distributed throughout the execution of the function. Both of these fields are entered as six-digit integer number.

2.5.1.3.5 Model Operation

The operation of PROTIME II consists of four logical phases. First is the input data translation and storage performed by the input preprocessor. Next the execution model initializes simulation parameters and calculates static simulation parameters based on the input data. Following this the simulation loop executes for the specified time. Lastly the output statistics are generated and the data placed in the output file. A HIPO chart representing the model is shown in Figure 2-15.

2.5.1.3.5.1 Preprocessor Operation

The PROTIME II preprocessor reads the input data and generates a copy for the print data set. An entity is assigned to each processor, device, function, internal event or external event as they are encountered in the input data. As the data associated with these entities is read it is converted to a basic unit (seconds, KIPS, KOPS, Kbytes for core or bytes and bytes/s for device transfers) and stored in the appropriate indexed array entry for the entity. When function linkage specifications are encountered they are converted to the appropriate pointers and stored in the indexed array entries associated with their entities.

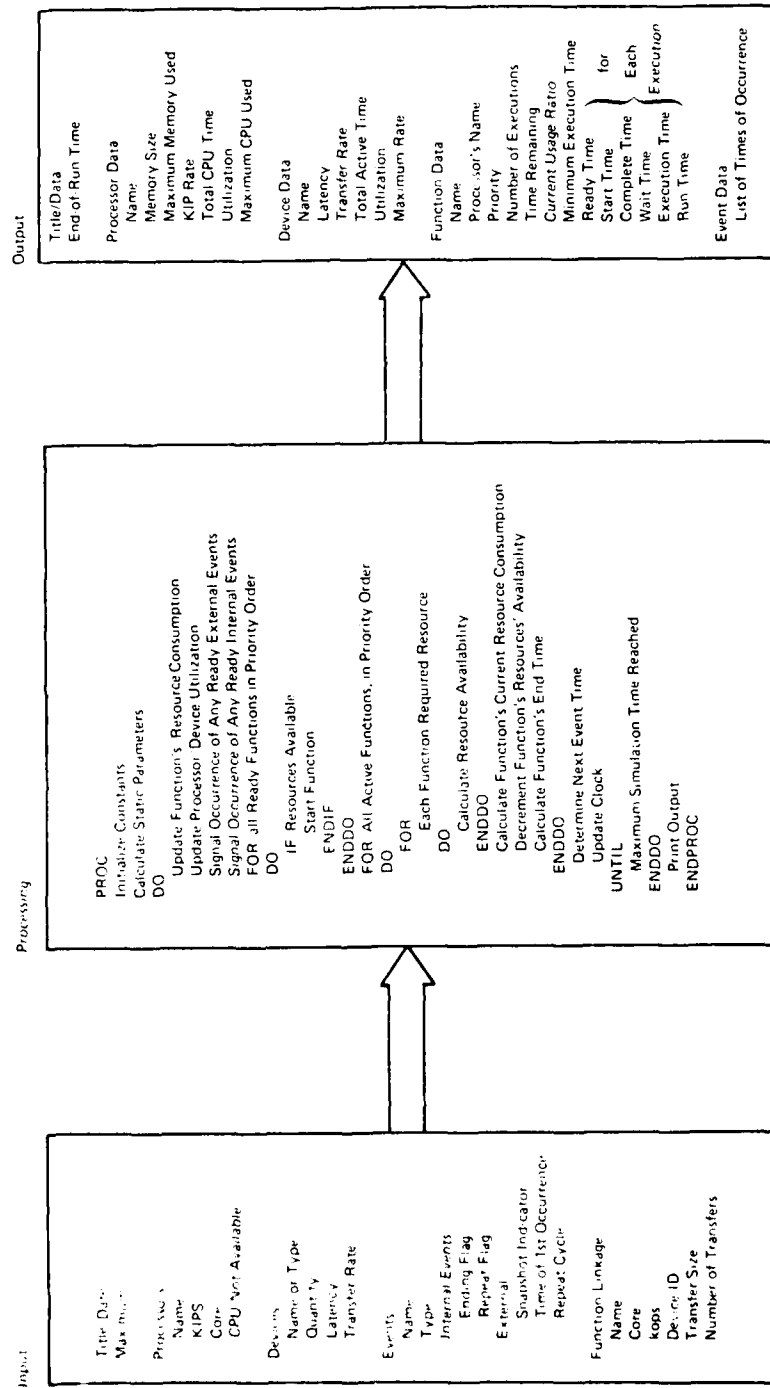


Figure 2-15. PROTIME II Data Entry Format

During data input to the data parameters are checked to verify that they conform to the proper format and are consistent with other input parameters. If any errors are detected an error message is printed and the execution terminated at the end of the data input phase.

2.5.1.3.5.2 Initialization and Static Parameters Calculation

The first task performed in this phase of the model is to initialize model constants. Following this, limited error checking, to verify that sufficient entities to accomplish a simulation have been specified, is performed on the input data and warning or error messages printed. If a severe error is detected in the data further execution is suppressed. The last task performed in this phase of the model is calculation of the static parameters, for each of the functions, as listed below:

1. Minimum processor time
2. Minimum device time
3. Minimum execution time
4. Function's device requirement
5. Function's device requirement
6. Function and event input (precedence) requirements.

The minimum process time, minimum device time, and minimum execution time are computed assuming no resource contention. Contention will be factored into the function's execution as the simulation executes. In addition, the time-line sequence of state changes based on event input data is established. During the calculation of these parameters the input data and calculated data are printed in a model entity format (see output description section).

2.5.1.3.5.3 Simulation Execution

The simulation execution phase of the model operates in a loop that continues until maximum simulation time specified in the input data is reached. During the execution of this loop if one of a limited set of errors is detected the loop is terminated and the data output phase is initiated.

At the start of each execution of the loop the next event is checked to see if it is a snapshot event. If it is, the required data is printed and execution then continues at the start of the loop. If no snapshot output is required the processor and device resource utilization and function execution time parameters are updated to reflect the effects of time and resource consumption based on the previous state of the system.

Next any external events that are ready to occur are processed by signaling their dependent functions and events and updating the time line. Following this any internal events that have satisfied their input requirements are processed in a similar manner. The ready function queue is then checked. The ready function queue contains functions that have been signalled to start but have not been previously able to obtain sufficient system resources to start execution.

All members are checked, in priority order, and if the required resources are available for any of them, they are made active. A new system state is then analytically calculated based on all active functions' resource requirements, priorities, processor resource availability and device resource availability. Following this the time at which all active functions will complete is recalculated for the new system state and an event placed on the timeline for the function that will complete first. Lastly the clock is updated to reflect the time that the next most imminent event will occur and control is returned to the start of the loop.

2.5.1.3.5.4 Simulation Output

The data output phase of the model is initiated following either a normal or error induced end of the simulation execution phase. The resource use and maximum instantaneous requirements for the processors and devices are calculated and printed. Next the functions' activities are traced by printing the function start, activation and completion times along with calculated wait and active times for each execution of the function. Lastly, each time of occurrence for each external and internal event is printed. The model processing is then terminated through the normal FORTRAN program exit.

2.5.1.3.6 Output Data Description

The model output description will be divided into four sections. First is the output of the preprocessor. Second will be the output of the input data and static parameter data in model entity format. Next is the normal model output generated during the model output phase. Lastly is the description of the optional snapshot output. The output listing from a system test run of PROTIME II is included in Appendix B and can be referred to while reading the next three sections.

2.5.1.3.6.1 Preprocessor Output

The primary output of the preprocessor is a listing of the input data in the same format as it exists in the input data set. This format is shown in Figure 2-14. If any errors were detected during the data input processing, a list of the errors will follow the input data listing. A sample of this output is shown in Appendix B, pages B-1 and B-2.

2.5.1.3.6.2 Input Data and Static Parameter Output

The input data and static parameters for the entities defined in the model are printed in model entity format. This output is shown in Appendix B pages B-3 to B-8. The first page has just the title (not printed in this system test output) and the maximum simulation time. The second page (B-4) contains the processor data and the third (B-5) the device data. The maximum CPU column for each processor is based on the percent unavailable entry in the user input. The next two

pages (B-6 and B-7) show the external event, internal event data and function data. Precedent events and functions are inputs to the indicated event or function and dependent events and functions are the events and functions receiving the signal output of the indicated event or function. The last page (B-8) contains a list of the elements on the external event queue and ready function queue.

2.5.1.3.6.3 Simulation Statistics

The normal model output includes processor and device resource utilization statistics, maximum resource requirements for the processors and devices, function execution time data, event times of occurrence, and warning messages generated during execution.

During simulation execution, if any conditions that may indicate invalid system operation are detected, an appropriate warning message is printed. These invalid conditions consist of errors in function and event linkage that make it impossible to assure correct signaling between the events and/or functions.

Following this title (not printed in this system test), date, run time, and processor data are printed as shown on page B-9. The "CPU available" is the CPU available at the end of the run. The maximum CPU is the maximum CPU utilization calculated during the run.

The device output data is shown on page B-10. The maximum rate is the highest device consumption (fractional) calculated during the run. The rate available is the device rate available at the end of the run.

The function output data is shown on page B-11. The minimum time is the no-contention execution time calculated at the start of simulation. The wait time is the start time minus the ready time, the execution time is the complete time minus the start time and the run time is the complete time minus the ready time. The ready, start, complete, wait, execution and run times are printed for each execution of the function. Lastly each times of occurrence for the external and internal events are printed as shown on pages B-12 and B-13.

2.5.1.3.6.4 Optional Snapshot Output

The ability to select snapshot output of selected parameters is available as a user option. Four sets of snapshots are available: processor, function, devices, and queues. These snapshots may be intermixed as desired, except that the processor snapshot, which is a subset of the devices snapshot, is mutually exclusive with the devices snapshot. The content of each snapshot is described below.

The processors and devices snapshots are intended for use both as model definition debugging aids and to gather a "time lapse" picture of resource use. The function and queues snapshots are intended primarily as model definition debugging aids.

The snapshots are created by the user as special external events. Thus, they can be specific to occur whenever required during the

simulation and can be requested to occur on a cyclic basis. Discretion should be used in requesting snapshot output because it can increase model run times excessively if used too liberally.

The processors snapshot prints the simulation time, next event time, next function time, head of the external event queue, time since last processor snapshot, and, for each processor, the processor name, current CPU available, maximum CPU available, total use to the present time, use since the last snapshot, and maximum memory used, each time it is executed.

The devices snapshot prints all the information listed above for the processor snapshot. In addition, it prints the time since the last devices snapshot, followed by, for each device, the name, current rate available, total utilization to the present time, use since the last snapshot, and the maximum rate required, each time it is executed.

The function snapshot prints the time, next event and function time, the next external event, and time since the last function snapshot. Next, the function's name, function's processor name, number of executions, remaining time, last start time, and current usage ratio are printed for each active function. Following the active functions, the function name, function processor's name, number of executions, memory required, and last function ready time are printed for each function on the ready function queue. Lastly, the internal event name, number of occurrences and incomplete (inputs expected but not yet received) external event, internal event, and function counts for each active internal event are printed.

Each time the queues snapshot is requested, the time, next event, and next function time are printed as a header. Following this, all the entities on the ready function queue, active function queue, external event queue, and internal event queue are printed.

2.5.2 RELIABILITY MODELING

In keeping with the definitions of Section 2.5.1, a reliability model is an abstract characterization of how a system responds to failures. The measures taken of such characterization are typically probabilistic numbers related to how successful the system is in terms of continued operations.

In contrast to the performance models, reliability models are primarily mathematical. Consequently, the process of reliability modeling a system consists of three steps — construction (in a mathematical sense) of a description of the failure modes and their effects in the system, measurement or estimation of values for the parameters making up the model, and calculation of the desired reliability statistics. The following sections describe IBM's approach to the first and the last of these. The actual measurement and estimation of parameters is covered in other sections such as 2.2 and 2.3.2.

2.5.2.1 Mathematical Model Fundamentals

The mathematical model to support analysis of system reliability is basically a probabilistic description of events relating to system failure, failure detection/correction, system reconfiguration, and system mode/state. The major elements of such a model are as follows:

1. A probability distribution for these events, with special emphasis on the failure activity since this set of events must always be included in the model.
2. A representation of the event space for this probabilistic description that efficiently and conveniently supports the generation of desired analytical results (such as mission reliability, MTBF, availability, etc.).
3. Mathematical or structural features to accommodate conditional dependence or time dependence of events and/or the nature of the probabilistic model.

For ATAS-like systems there are at least four specific system activities that must be included in the preceding model elements:

1. Error Detection - Most real systems will not detect 100% of all errors.
2. Retry - Many failures are transient, and simply retrying the disturbed operation is often a meaningful response to detection of a failure. Again, however, this is not 100% effective.
3. Isolation - Given that a system cannot recover from a certain failure by retrying, it may seek to recover by indentifying the faulty subsystem and working around it. However, as with error detection, fault isolation is seldom 100% accurate, and a good reliability model must take this into account.
4. Reconfiguration - Given that a subsystem has isolated a fault, it may try different techniques to recover from it. These techniques have varying degrees of success, and usually affect the ability of the system to withstand future faults.

Section 4.1 describes these four activities in more detail.

As might be expected, the particular model one selects for reliability analysis is strongly a function of physical failure mechanisms and system operation features. The following discussion provides a review of the principal reliability modeling methods which have been developed.

A variety of failure-distribution models may be used to accommodate the absence or particular nature of "wear-out" mechanisms in the components of a system. The most common of these is the exponential distribution, which corresponds to a single-parameter, constant-failure-rate condition. The use of this model essentially presumes the absence of a wear-out mechanism during the operational life period of the components used in a system. Since this condition is reasonably well met in the operation of most solid-state electronic systems, the exponential or Poisson failure distribution model is commonly used in the reliability analysis of such systems. The use of such a model usually allows one to

generate analytical results for system mission reliability, MTBF, or availability with relative ease. Analytical complexity increases significantly as one is forced to use more complex failure distribution models.

A number of the models may be used to describe failure distributions as the wear-out phenomenon takes a more complex nature than that commonly encountered in solid-state electronic systems. For example, a single parameter Rayleigh distribution may be used to model components that feature a linearly increasing failure rate throughout the system operational life. A Weibull distribution with shape and scaling parameters may be used to characterize a variable failure rate common in electromagnetic components with moving parts. Note that the Weibull distribution may include the exponential and Rayleigh distributions as special cases when the scaling and shape parameters are appropriately fixed. Another distribution model that can be used to replace the Weibull model for systems with variable failure rate components is the Gamma distribution. The necessity to use these more complex failure distribution models is frequently avoided by modeling system components at a sufficiently low level that the exponential failure distribution may be assumed. The generation of system reliability, MTBF, and availability analysis results for Rayleigh, Weibull, and Gamma models involves considerably more complexity than that for exponential failure distribution models.

A second major element of the reliability model is the definition of the event space for the failure, failure detection/correction, system reconfiguration, and mode/state activity. For relatively simple systems, these may be described in terms of truth tables or Boolean equations that can be conveniently mapped to probability distribution equations. In more complex structures, system mode/state diagrams are necessary to identify mode/state event sequence and the conditions that cause mode/state changes in the system. Graph theoretic methods for both tracing, subgraph decomposition, and cut-set/tie-set analysis can then be used to determine the event space of a set of states without enumerating them. The use of these techniques then permits one to reduce the complexity of the event space model to a point where analysis for a complex system becomes tractable.

In the definition of the failure distribution model and an event space model for a particular system and its operating scenario, one must construct these in such a way as to include time and event dependence features. For relatively simple systems, except for failure events, the probability model may be time independent and structurally represented in terms of series/parallel block diagram. On the other hand, for systems that are reconfigured to effect repair of a constant failure rate component set, a discrete state/time dependent Markov process model will typically be used. A more complicated discrete state/time dependent Semi-Markov process model is used to accommodate structures with variable failure and repair rate properties. Simulation models may be required to support more realistic and detailed

representation of selected parameters in the system model for very complex structures.

The models that will be used for performing reliability analysis and trade studies for ATAS design are based on the assumption that the failure and fault processes of the system elements are Poisson during the mission time period. This is a reasonable assumption since failure rates or fault rates can be considered to be constant for the short time periods of the missions being analyzed. Consequently analytical models that are based on failure processes or failure distributions such as a Weibull, Gamma, Rayleigh, Normal, etc., will not be used.

Given that λ represents a fault rate, then a Poisson process is defined as follow:

1. The probability of a fault (failure) in the interval $(t, t + \Delta t)$ is $\lambda \Delta t$.
2. The probability of more than one fault in the interval is zero.
3. The probability of no faults occurring in the interval $(t, t + \Delta t)$ is $1 - \lambda \Delta t$.

Note: Δt is a suitably small increment of time, such that at most 1 failure can occur in the interval.

Mission reliability functions will be defined which express the mathematical relationship of the system relative to the modules or functional groups that make up the system.

For this study, two representative mission scenarios were postulated for use in defining mission profiles and operational requirements for each mission phase. (See Reference 3). Avionics functional requirements for each mission phase were developed from these two scenarios. Operational reliability block diagrams were then developed from the baseline avionics equipment and avionics functional requirements for each mission phase. The F-15A avionics equipment list was the baseline for determining equipment use in the primary and back-up modes for satisfactory mission completion. The reliability model used was as follows:

1. The primary and back-up modes consisting of the major functional groups (such as navigation and guidance, fixtaking, etc.) for each mission and each phase of the mission were defined.
2. The hardware elements of the system were assigned to the appropriate functional groups.
3. The failure rates of each of the functional group blocks was determined based upon the summation of the failure rates of the parts.
4. The reliability of each block for each mission and mission phase was determined.
5. The reliability for each phase for each mission was calculated using the follow model.

$$R_m = \prod_{i=1}^m R_{fi} + \sum_{i=1}^{NI} (1 - R_{fi}) \prod_{\substack{k=1 \\ k \neq i}}^{LK} R_{fk}$$

- where
- 1) R_{fi} is the reliability of functional group i
 - 2) m is the number of functional groups
 - 3) NI is the number of back-up modes
 - 4) LK is the number of functional groups in the back-up mode.

The first half of the above equation represents the prime mode, while the second half represents the contributions of the back-up modes. The current analysis of the two postulated mission reliabilities employed this model, and it will be used for computing mission reliabilities during Phase II.

There are two other levels of reliability modeling required for the Phase II study. One is to determine the impact on the reliability and failure rate of system elements making up a computer network configuration, as a result of the addition of new error-handling techniques. Many such techniques are being evaluated, and each case must be considered individually. As one example, consider the case of adding an error correcting code to memory.

For a monolithic memory to store 32-bit data words, the failure rate of the hardware associated with one word might be:

$$\lambda = 32 \times 10^{-9} \quad \text{failures per hour}$$

To store N words, and if there is no fault tolerance, the reliability of this hardware would be:

$$R_M = (e^{-\lambda t})^N$$

$$\begin{aligned} \text{For example, if } N &= 4096 \text{ words} \\ t &= 1000 \text{ hours} \end{aligned}$$

$$R_M = 0.877 \quad \text{Probability of success}$$

If a single error correcting code is provided, then 38 bits would be required to store the data and the code, but any single-bit fault could be tolerated.

Then the reliability for one word would be:

$$R_w = e^{-38 \times 10^{-9} \times t} + 38e^{-37 \times 10^{-9} \times t} (1 - e^{-10^{-9} \times t})$$

and the reliability of the memory is:

$$R_M = (R_w)^N$$

$$R_M = 0.99997, \text{ probability of success for 1000 hours.}$$

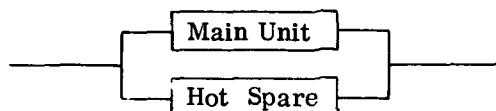
This model is a mathematical equation which relates the probability of success for the memory in terms of the probability of occurrence of one or less bit failures in each of the 38-bit words which makes up the single-fault-tolerant memory.

The third level of reliability modeling is required to evaluate various options for redundant modules in the computer network configuration. Three specific cases will be considered, as follows:

1. One or more hot spares
2. One or more cold spares
3. No spares, but fall back to degraded capability.

This analysis must include the effect of nonsuccessful recovery (i.e., there is a probability that the reconfiguration process will not be completed successfully, due to incorrect or incomplete error isolation, or a failure in the configuration switching mechanism). Again, these cases must be looked at individually. Some examples of how this analysis might proceed are given below.

If the processor is to be made fault tolerant by means of adding a second processor (a hot spare), the analysis would be as follows:



Let λ be the failure rate of one processor. Then the probability of success of each processor is $R = e^{-\lambda t}$, where t is the mission or phase time. The probability of success (to have at least one processor operating at the conclusion of the mission or phase) would then be:

$$P_s = R^2 + 2R(1-R) \\ = 2R - R^2$$

This design would have to be such that failure of one of the processors would not result in a failure of the two processor system; i.e., the probability of failure detection and recovery is equal to one.

In the event that the second processor is to be a cold spare and the probability of detection of failure of the main processor is less than one, then the analysis would be as follows:

Assume two processors, each of which is capable of performing the job, and where one is operational and the other is an unpowered standby spare.

If:

V is the failure rate of the operational unit

U is the failure rate of the standby unit

T is the mission time

$$V \geq U$$

C is the probability of detecting the failure of the operational unit and switching in the spare.

Then the probability of success is:

$$P_s = e^{-(V+U)t} + \frac{CV + U}{U} e^{-VT} (1 - e^{-UT})$$

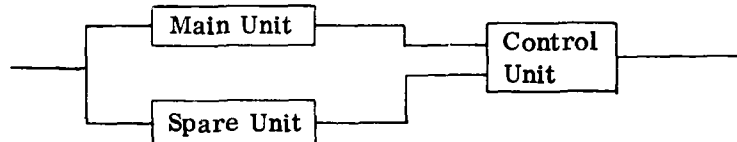
If both units are operational but the main unit is "driving" the system and the probability of detecting the failure and turning control over to the backup unit is less than one, the probability of success would be as follows:

$$P(s) = e^{-2Vt} + (C+1)e^{-Vt} (1 - e^{-Vt})$$

Note that if C is equal to one this design would be equivalent to the previous configuration. (Since both units have power on, the failure rates are the same.)

If the design is to have an active set of system elements and a standby set of system elements with an interface control unit, the analysis would be as described below.

Assume two system elements (one active and one standby) and an interface control unit (e.g., a unit that will switch when the main unit fails).



$$R = R(v) \left[1 + \frac{CV}{U+I} (1 - R(U+I)) \right]$$

where:

V = The power-on failure rate of the main or active system elements.

U = The power-off failure rate of the standby system elements.

I = The power-on failure rate of the interface control unit.

C = The probability of recovering from a failure of the active set of system elements, given that the interface control unit is operating and the standby set of system elements is capable of operating

$R(v) = e^{-Vt}$ is the reliability of the active system elements for mission time t.

$R(U+I) = e^{-(U+I)t}$ is the reliability of the interface control unit and the standby system elements for mission time t.

Another approach to fault tolerance could be to achieve it by means of reduced capability. For example, assume that the design is to have a system with six or seven elements (which might not necessarily

be identical). If one of the system elements were to fail, its job would be performed by the remaining elements of the system. For this to work, non-critical tasks would have to be dropped or would have to be performed less often. For this situation, the analysis would be as described below.

Assume seven identical system elements, which would provide successful operation if at least six elements were operating. The probability of success would be:

$$\begin{aligned} P(s) &= R^7 + 7R^6(1-R) \\ &= 7R^6 - 6R^7 \end{aligned}$$

where R = The probability of success of each of the system elements for the mission time t .

In summary, no single "equation" or approach to developing reliability equations is universally usable. Different kinds are required for various situations in the Phase II study, and will be developed as needed.

2.5.2.2 Model Implementation

To summarize the previous section, three different levels of reliability models will be developed:

1. Computer module reliability as a function of new error-handling techniques such as ECC.
2. Computer network reliability as a function of redundancies, spares, degraded modes, etc. Note that these degraded modes are different from mission degraded modes.
3. Mission reliability (system reliability).

For each of these, the relevant equation must be derived, parameters estimated, and reliability results computed. Further, the models are applied sequentially, with statistics from one model used to drive the next. Thus, for example, reliability of a memory using ECC is produced by a module reliability model, and used in the computer network model. The output of this is then used in the mission model.

Implementation of a model refers to the whole process of deriving equations and computing statistics. For the computer module level of modelling, the derivation must be done by hand, because the fault tolerant techniques used for each module are usually unique. For example, the equations for a memory with ECC are entirely different than those for a CPU with self-checking logic. Likewise, the relevant statistics are computed only once, or at best a very few times, negating the need for any sophisticated computer program aid. The kinds of statistics computed include such things as failure rate, coverage, transient error rate, etc.

The third level of modelling, mission reliability, is equally straightforward. The equation was given in the previous section, and uses data derived from the second level models.

The most complex modelling task is that of the second level, namely computer network modelling. To construct a good and accurate model at this level requires consideration of parameters such as

1. Failure rates of individual modules.
2. Coverage for individual modules (percent of faults detectable).
3. Transient fault arrival rate of individual modules.
4. Duration of transient faults.
5. Number of modules needed of each type for full performance.
6. Number of spares of each type.
7. Sequence of spare usage, and what "degraded modes" are available after exhaustion of all spares. "Degraded modes" in this sense includes a reduction of fault tolerance or future system reliability, not simply performance losses.
8. Characteristics of retry operations (how long, how many, how effective, etc.).
9. Characteristics of isolation and reconfigurations.

In addition to all these parameters, another complicating factor is the obvious desire to try a possibly wide range of radically different configurations making up the computer network. Although it is possible to derive expectations for each such configuration, the work involved is quite time consuming and error-prone. It would be of obvious benefit if this process could be mechanized, perhaps by some computer assists.

A variety of such computer-assist packages exist, including CARE, RELCOMP, ARIES, etc. Of these, one that looks particularly useful is ARIES by Ng and Avizienis at UCLA. (See References 8 and 9.) This package is a set of APL programs that models computer systems consisting of sets of homogenous subsystems. The user provides at a relatively high level all the parameters described above (plus repair facilities if desired). ARIES then constructs automatically a reliability model based on Markov chains. The user may then ask of the package such questions as reliability as a function of time, MTBF (mean time between failures), MTTF (mean time to first fault), mission time at a given reliability level, normalized probability of failure for each subsystem (i.e., how much failure rate does each subsystem contribute to the total system), etc. Based on sample runs made to date at IBM Owego, the package appears both easy to use and quite fast.

The one problem with ARIES is that within a "subsystem" all spares and all active modules are very nearly the same in terms of reliability statistics. This presents a problem when, for example, the network consists of a set of identical processing modules each having different amounts of memory.

A spare module can be switched in for any of the operational ones. However, the spare is assumed to have an amount of memory equal to the largest found in any of the active units. Because memory contributes a large part to the failure rate, the failure rate for active and spare processing modules will be different. A truly accurate model should take this into account, but ARIES does not.

To get around this two separate approaches will be used. The first is to use ARIES as is for the entire network, but assuming for each subsystem that all modules have characteristics equal to that of the largest. This will give a lower bound to the computer network reliabilities.

For those candidates that look most promising, a more detailed model will be developed in a two-step fashion. First, ARIES will be used separately on each subsystem for which it is directly appropriate, and reliability as a function of time computed. This factors in all the transient failure rates, coverage, etc. Then through use of algebraic techniques such as described in the previous section, these reliability vectors will be multiplied, added, etc., as required to model the entire system. The result is a numerical table giving total computer network reliability as a function of time.

2.5.3 LIFE-CYCLE COST MODELING

This subsection of the report documents the activity and findings involved in the determination and description of the life-cycle cost (LCC) model most suited for the accomplishment of LCC trade studies that will evolve during the design and development of a distributed computing system for the advanced tactical avionics system (ATAS).

2.5.3.1 Candidate LCC Models

Four LCC models were reviewed for applicability and possible consideration for use. A brief description of each is provided below.

2.5.3.1.1 Logistic Support Cost Model

Developing Activity: Air Force Logistics Command, Acquisition Logistics Division

Users Handbook dated: August 1976

Prime Service User: U.S. Air Force

Model Language: FORTRAN IV

Structured Levels of Maintenance Concept: Flight Line, Base, Depot

Type of Model: Deterministic

Availability for use: Available

Analysis of Complexity-Completeness: Ten basic, relatively independent equations are used of which eight may be used for avionics. The other two, fuel consumption and spare engines, are optional and special purpose. The model includes those support functions which traditionally exhibit the most impact to LCC. The subject of the remaining eight cost equations follows:

1. Initial and replacement LRU (line replaceable unit) costs
2. On-equipment maintenance costs
3. Off-equipment maintenance costs
4. Inventory entry and supply management costs
5. Costs of support equipment

6. Costs of personnel training and training equipment
7. Costs of management and technical data
8. Facilities costs.

LRU spares computation is based on expected backorder performance or probability of stock outage. Spares quantities for lower level parts (SRUs, piece parts) are not considered. It is an accounting type model which is sensitive to program and hardware design characteristics. Its primary use is for the evaluation of design alternatives before the details of the hardware are known. All using activities (operational sites) are considered identical.

2.5.3.1.2 Spares Optimized Inventory Level Selection (SOILS)

Developing Activity: IBM - Federal Systems Division

Users Handbook dated: September 1979

Prime Service User: U.S. Navy (Shipboard Deployed Helicopter)

Model Language: FORTRAN IV

Structured Levels of Maintenance Concept: Organizational (Shipboard, or land-based), Intermediate (Intermediate Maintenance Activity), Depot

Type of Model: Deterministic

Availability for Use: Available

Analysis of Complexity-Completeness: SOILS is designed to develop a cost effective set of spares to support a series of single systems in remote environments and meet an availability requirement. LCC equations including those which depict the impact of design/hardware changes, were added.

The subject of the cost equations follow:

1. Initial WRA (weapon replaceable assembly) spares
2. Initial SRA (ship replaceable assembly)/piece part spares
3. Organization level labor
4. Normal intermediate labor
5. Intermediate BCM (beyond capability of maintenance) labor
6. No defect labor
7. SRA repair labor
8. Depot BCM (condemnation) labor
9. Replacement WRA/SRA costs
10. Replacement piece parts costs.

It considers a system of many WRAs with a fractional consideration for SRAs. Its primary use is in its ability to select WRA spares to meet an availability criteria and for the evaluation of hardware, operational system and support system changes.

2.5.3.1.3 Life-Cycle Cost Model

Developing Activity: Naval Material Command, Naval Weapons Engineering Support Activity

Users Guide data: January 1977

Prime Service User: U.S. Navy

Model Language: SIMSCRIPT 2.5

Structured Levels of Maintenance: Organizational, Intermediate (IMA), Depot

Type of Model: Deterministic

Availability for Use: Program is available but is used with extreme difficulty since the SIMSCRIPT 2.5 language is not supported in this facility.

Analysis of Complexity-Completeness: This program is very complete and, therefore, complex. It encompasses R&D, investment, and operating and support costs as major cost elements. These are subdivided into 85 cost elements of which 61 are basic equations. These equations utilize combinations of 104 input factors.

Spares are calculated through a complex equation involving the failure frequency, stockage times, duty cycle, reliability improvement/degradation factor, and others. Wide flexibility in operating scenario is provided the user at the expense of increased complexity.

2.5.3.2 Program Life-Cycle Cost

Developing Activity: Analytical Sciences Corporation

User's Guide Dated: Undated

Prime Service User: Appears to be Air Force

Model Language: FORTRAN IV

Structured Levels of Maintenance: Flight Line, Base, Depot

Type of Model: Deterministic

Availability for use: Available

Analysis of Complexity-Completeness: This is a medium complexity model encompassing some fifteen cost equations with others to determine spares, support equipment, etc. It utilizes approximately 74 input factors to calculate a full range of initial and recurring logistic cost elements. Spares are computed based on cost effectiveness to reach an equipment availability requirement.

The subjects of the 16 cost equations follow:

1. Initial training
2. Data acquisition
3. Item entry
4. Data management
5. Prime hardware
6. Support equipment
7. Initial spares
8. Installation
9. Warranty
10. Flight line maintenance
11. Base level maintenance
12. Depot level maintenance
13. Item management

14. Data management
15. Packaging and shipping
16. Support equipment maintenance.

It can use three levels of hardware indenture; system, LRU, SRU. The model could be used for tracking, but would probably be more effective at analyzing alternate hardware configurations, trade studies, and the effect of ECPs. It includes the capability to analyze the effectiveness of reliability improvement warranty programs and could be used to test the advantage of various maintenance concepts.

2.5.3.3 Analysis of Models/Selection of Application Model

Although each model implements an operational profile with different techniques, any one of the models could represent the avionics system of an advanced tactical fighter. Of greater concern is the detail with which each model determines the acknowledged high LCC drivers, i.e., initial and recurring spares, test equipment and the support thereof, and training. Also of prime consideration is the availability and workability of the models.

The relative advantages and disadvantages of using each model were assessed. Models 1, 2, and 4 are written in FORTRAN IV computer language and are presently available for use. Model 3 is written in SIMSCRIPT 2.5 and, although it could be made available, is not presently being used because it is difficult to work with. To adjust model 3, a Navy model, to an Air Force maintenance concept in an unfamiliar language could prove time consuming.

Model 1, although it is based upon the Air Force maintenance concept, was determined to be of insufficient detail for this application. Model 2 selects initial spares very well, but does not, as yet, contain the full complement of logistics cost elements.

Table 2-30 summarizes the decision criteria by model.

TABLE 2-30. DECISION CRITERIA BY MODEL

	Model Language	Availability For Use	Need to Modify	Complexity Completeness	Basic Equations
1. LSC Model	FORTRAN IV	Good	No	Medium	8
2. SOILS Model	FORTRAN IV	Good	Yes	Medium	10
3. LCC Model	SIMSCRIPT 2.5	Difficult	Yes	Very Complex	61
4. Program LCC	FORTRAN IV	Good	No	Medium	15

Model 4, Program LCC, was selected as the model for this application since it is currently available, is of good detail for trade study analyses and includes all of the high LCC drivers. It can be used for a range of sensitivity analyses on any grouping of its parameters.

Since development costs are not an integral part of any of the above models, it will be necessary to add them to the selected model.

2.5.3.4 Selected Model

Having selected the application model, the following sections contain further descriptive information concerning program LCC. Section 2.5.3.3.2 describes the logical steps that the program processes through to compute the costs and create the output reports. Section 2.5.3.3.2 describes the simple sequence an operator implements to execute the program. Section 2.5.3.3.3 contains a brief description of the 54 system-related input parameters and 20 hardware-related input parameters which are used in program LCC. A sample output report has been included as Appendix C and may be used for reference with the following sections.

2.5.3.4.1 Functional Sequence in Program LCC

Program LCC operates functionally similar to many other LCC analysis programs. The following steps describe the major functions that program LCC performs in its analysis of the input data:

1. A date is created from within the host computer for placement on the first page.
2. The first line of the data set is read. This series of eight "1s" or "0s" constitute the print command for the eight optional tables for printout. If no tables are requested the program is halted. The eight optional tables are identified and discussed later.
3. The remainder of the data set is read in. This includes all classes of data described in the program guide such as the standard cost factors, logistic factors, hardware definition, support equipment data, and contractor data. These categories of data will be further described and detailed later. Some preliminary checks of the data are made, i.e., you can't plan on repairing an SRU at base level while planning to repair its LRU at the depot level. A crosscheck of the number of operational systems by overseas base type and CONUS base type are checked against the total numbers of systems. If a disparity exists, the program is halted.
4. Selected input parameters may now be interactively changed, if so desired.
5. Sensitivity analyses may be interactively initiated by selecting the specific parameter, then inserting its upper limit, lower limit, and the number of steps desired.
6. If the first element of the print command is a "1" then optional table 1, the input data, is printed out.

7. The monthly reliability growth is computed from the yearly reliability growth data.
8. Monthly removals by equipment are calculated and then the number of periodic maintenance cycles.
9. The program does some validation of input data at this point. It checks that the system and shop availability parameters are between zero and one. Of the hardware data, the equipment failure verification must occur before (or at a lower maintenance level than) that at which the equipment is repaired. The unverified failure probability (function of false removal rate), the condemnation rate, and the "not repairable this station" rate are all checked to be between zero and one. Selected other parameters are checked for reasonableness. Failure to pass these validation checks causes an error message to be printed and the program is halted.
10. The requirements for support equipment are established based upon the desired level of repair for each equipment, the computed working hours per month and the associated support equipment. Support equipment quantities are determined for each base as well as the depot.
11. If the print command so requires, the support equipment requirements are printed as option 2.
12. Depot spares, base spares, and condemnation spares are calculated. Depot and condemnation spares quantities are calculated for each LRU and SRU. LRU spares are calculated by base type.
13. If the print command requires, a print out is made of the spares data either in a detailed manner, option 3, or summarily, option 4.
14. Manpower requirements are computed based upon task frequency and the time to perform the task. All three levels of maintenance are considered for this computation.
15. If the fifth print command, option 5, has been initiated the manpower requirements by maintenance level are printed.
16. Acquisition costs are then computed. These cost elements are; initial training, data acquisition, item entry, data management, prime hardware, support equipment initial spares, installation, and, if used, warranty.
17. Operating and maintenance costs are next. These cost elements are; flight line, base level and depot level maintenance, item management, data management, packing and shipping, and support equipment maintenance.
18. If the sixth print command has been implemented, the undiscounted and present value costs for the cost elements computed in 16 and 17 above, will be printed.
19. If the seventh print command has been implemented, the undiscounted and present value costs for each year of the operational life of the system are printed.
20. If sensitivity analyses have been requested in step 5 above, and if the eighth print command has been implemented, the selected

parameter is computed through its selected range. Sensitivity analysis printout will contain the parameter name, each value tested and its associated undiscounted and present value costs.

Once the program has progressed to the sensitivity analysis, or through them if they have been selected, the program is complete and operation is halted.

2.5.3.4.2 Sequence to Operate Program LCC

Operating program LCC as implemented in IBM's CMS System 370/168 is very easy. The data set is first verified or changed as desired. The operator calls the implementing executive program, called GOLCC, followed by the name of the selected composite data set, which again is followed by an arbitrary name used to entitle the output listing. The operator is asked if he desires to interactively adjust program parameters. If not, the operator is asked if sensitivity analyses are desired. If so, the selected parameter is identified along with its range of values and number of steps. If sensitivity analyses are not desired or upon responding with the previous information, the program runs to completion. Copies of the output listing can then be printed. A brief sequential flow diagram of the above process is included as Figure 2-16.

2.5.3.4.3 Program LCC Input Parameter Identification/Definition

There are 74 parameters used in program LCC. These parameters are functionally divided as follows:

1. Multipliers for Cost and MTBF (2)
2. Standard Elements File (11)
3. Logistic Factors File (22)
4. Hardware Definition File (20)
5. Support Equipment Definition file (4)
6. Contractor Data File (15)

The following is an identification of input parameters grouped by the functional divisions above. A further definition of the parameter is included where necessary for clarity:

1. Multipliers for Cost and MTBF — The original program was modified to include these two factors. Since the installed sensitivity analyses only change a single item's MTBF, cost, or other factor, to achieve a system impact of varying cost or MTBF, two multipliers of these parameters were added. They are also printed out to monitor their value.
 - a. Cost multiplier
 - b. MTBF multiplier
2. Standard Elements File Input Parameter Identification/Definition —
 - a. Item Entry Cost/New Item - Cost for entering a new item into the Government supply system.
 - b. Base Labor Rate/Hour - Standard base labor rate in dollars per manhour.

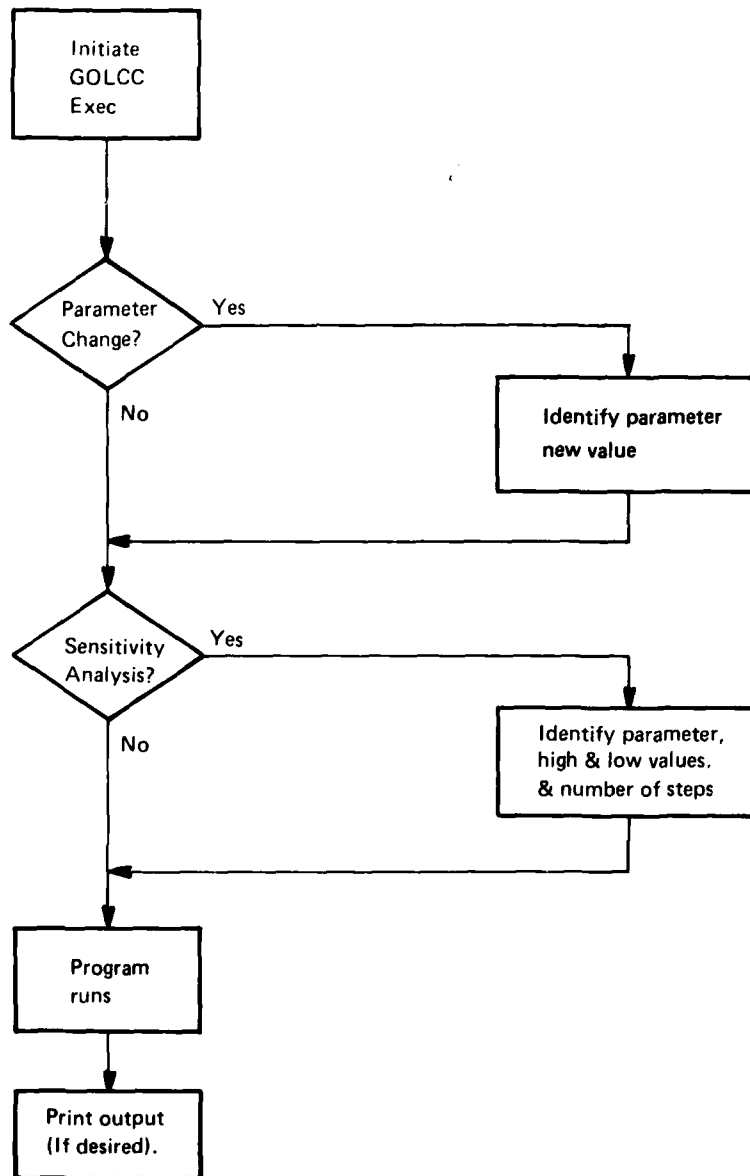


Figure 2-16. Program LCC Operational Flow

- c. Depot Labor Rate/Hour - Standard depot labor rate in dollars per manhour.
 - d. Packaging & Shipping Cost/lb. (CONUS) - Standard Cost in dollars per pound for packaging and shipping units between the depot and overseas bases.
 - e. Packaging and Shipping Cost/Lb. (Overseas) - Standard Cost in dollars per pound for packaging and shipping units between the depot and overseas bases.
 - f. Initial Data Management Cost/Copy/Page - Standard Cost in dollars per copy per page for reproduction and distribution of technical data.
 - g. Item Management Cost/Item/Year - Standard inventory management cost in dollars per year.
 - h. Data Management Cost/Page/Year - Standard data management cost in dollars per page per year.
 - i. Base Material Consumption Rate - Consumable materials rate in dollars per manhour at base level.
 - j. Depot Material Consumption Rate - Consumable materials rate in dollars per manhour at depot level.
 - k. Discount Factor - Annual discount factor applied to future costs.
3. Logistics Factors File Input Parameter Identification/Definition -
- a. Operational Life-System operational life in years.
 - b. Number of Bases (CONUS) - Total
 - c. Number of Bases (Overseas) - Total
 - d. Number of Systems (CONUS) - Total
 - e. Number of Systems (Overseas) - Total
 - f. Number of Unique Bases (CONUS & Overseas) - with the associated number of systems at that base.
 - g. Cost/System Installation
 - h. System Operating Hours/Month
 - i. Number of Depot Work Shifts
 - j. Base Resupply Time (CONUS) - Resupply time in hours between the depot and CONUS bases.
 - k. Base Resupply Time (Overseas) - Resupply time in hours between the depot and overseas bases.
 - l. Depot Replacement Cycle Time (Hours) - Depot repair cycle time in hours for unit which can be repaired by removal and replacement operations.
 - m. Depot Repair Cycle Time (Hours) - Depot repair cycle time in hours for units which require actions more complex than removal and replacement operations (NRTS type repairs - see paragraph 4m below.)
 - n. Shipping Time to Depot (CONUS) - Shipping time in hours to depot from CONUS bases.
 - o. Shipping Time to Depot (Overseas)- Shipping time in hours to depot from overseas bases.
 - p. Base Turnaround Time (Hours)

- q. System Warranty Period (Years) - Set to Zero if warranty is not planned.
 - r. Spares Objective (System) - System availability objective (the steady-state probability that an aircraft is not in NORS - Not Operationally Ready due to Supply - status due to an LRU backorder).
 - s. Spares Objective (Shop) - LRU availability objective (the steady-state probability that an LRU is not in an unrepairable state at base level due to a backorder on the SRU spare supply).
 - t. Depot Stock Safety Factor - Listed in standard deviations.
 - u. Activation Schedule - Sequential number of systems installed by month/year.
4. Hardware Definition File Input Parameter Identification and Definition - The abbreviation is that used in the model and is indicated in the printout. (See Appendix page C-2).
- a. Number of Replaceable Units - This is a slight misnomer. It is used in the program to indicate the number of line items of system, LRU and SRU which follow. This number must equal the number of lines or the program will halt.
 Note: The following are repeated for each included item (system, LRU, or SRU),
 - b. IN - Indenture of the item; 1 = System, 2 = LRU, 3 = SRU for the next previous LRU.
 - c. Nomenclature - Brief name or designation of the item.
 - d. NQ - Quantity of the item required per system.
 - e. CRU - Cost in dollars per unit for a spare of that unit.
 - f. MTBF - Mean-time-between-failures of that unit in hours. If MTBF degradation must be used, either the degraded values can be pre-calculated or the MTBF Multiplier listed in 1.b above can be used.
 - g. UFP - Expected fraction of removals of the unit that will be unverified failures. This fraction can be considered as a false removal (pull) rate. Applicable only to LRUs.
 - h. W - Weight in pounds of the unit.
 - i. FVS - Average time in hours for failure verification of the unit. Applicable only to LRU's.
 - j. RLS (System) - Average time in manhours for in-place system repair.
 - k. RLS (Units) - Average time in manhours for NRTS (not repairable this station) repair of the unit.
 - l. RRS - Average time in manhours to isolate a failure to the unit, remove it (include access), replace it with a spare, and verify the corrective action.
 - m. RMS (system) - Average materials cost for in-place system repair.
 - n. RMS (units) - Average materials cost for NRTS repair of the units.
 - o. NRTS - Expected fraction of failures of the unit that are repairable only at depot.

- p. COND - Expected fraction of unit failures resulting in unit condemnation.
 - q. LV - Maintenance level of failure verification:
0 = Flight line, 1 = Base, 2 = Depot.
 - r. LSEV - The line item of support equipment used for failure verification of the unit. Applicable only to LRUs.
 - s. USEV - Support equipment usage time in hours for failure verification of the unit. Applicable only to LRUs.
 - t. LR - Maintenance level of repair for the unit; 0 = Flight line, 1 = base, 2 = Depot.
 - u. LSER - The line item or items (up to 4) of support equipment necessary for repair of the unit.
 - v. USER - Support equipment usage time or times (up to 4 and related to u. above) in hours for repair of the unit.
5. Support Equipment Definition File Input Parameter Identification and Definition
- a. Number of line items of support equipment - Must equal the succeeding lines of included support equipment.
 - b. Nomenclature - The name or designation of the test set. Practically, it helps to have the functions of the test set included in the name.
 - c. Cost - The unit cost of the support equipment set. This cost must include all special tools or fixtures necessary for use of this equipment.
 - d. O&M Cost Factor - Annual cost to operate and maintain the set of support equipment. This is expressed as a fraction of the set cost in c. above.
6. Contractor Data File Input Parameter Identification and Definition
- a. Acquisition Cost/System (dollars/system)
 - b. Base Level Training Cost (dollars)
 - c. Depot Level Training Cost (dollars)
 - d. Data Acquisition Cost (Base Level Manuals) - Base repair technical orders cost (dollars)
 - e. Data Acquisition Cost (Depot Level Manuals) - Depot repair technical orders cost (dollars)
 - f. Data Acquisition Cost (other) - Operation technical orders cost (dollars)
 - g. Pages of Data (Base Level Manuals) - Base repair technical orders.
 - h. Pages of Data (Depot Level Manuals) - Depot repair technical orders.
 - i. Pages of Data (other) - Operation technical orders
 - j. Number of New Inventory Items - Number of new items (no Federal Stock Number assigned) in the proposed design which must be stocked by the Government to support system maintenance. Includes LRUs, SRUs, and piece parts.
 - k. Contractor Base Resupply Time (CONUS) - Average resupply

time in hours from contractor to CONUS bases. Applicable to warranty condition.

- l. Contractor Base Resupply Time (Overseas) - Average re-supply time in hours from contractor to overseas bases. Applicable to warranty condition.
- m. Contractor Repair Cycle Time - Contractor depot repair cycle time in hours. Applicable to warranty condition.
- n. Warranty Price - This is the contract price of the reliability improvement warranty or repair warranty. It must be associated with the system warranty period identified in the Logistics Factors file. This parameter is set to zero if warranty is not an issue.
- o. Reliability Growth Profile - This is a by-year ratio of the projected system MTBF to the initial MTBF.

2.5.3.4.4 Program LCC Output Format

Included as Appendix C are sample reports which can be printed after running this program. They are printed in the sequence described in Section 2.5.3.3.1 of this report. The eight options identified there have been added to further identify each report. Since this is intended to show only format the input data is not accurate and may not be applicable.

2.5.3.5 Summary/Conclusion

Program LCC is a very useful tool in evaluating different/related designs or modifications. It is a flexible tool in its ability to analyze alternate system maintenance concepts and various sensitivities. The only void which has been noted is that of software cost and software maintenance and support. This cost element, though in many cases very significant, should not impact the analysis of the ATAS application.

It is not anticipated that any of the system related input parameters will change as the Phase II study evolves requiring LCC impact analysis. It is estimated that three of the 20 hardware related parameters are most likely to change for this type of analysis. These are the cost (CRU), the reliability (MTBF), and the measurement of false removal rate (UFP).

Probably the most difficult task in LCC analysis is that of collecting data. To perform a system level analysis for the ATAS system would require that LRU and SRU data be available for each replaceable unit or subsystem in the F-15 aircraft avionics system. In the event that such data is not readily available, one of the following two courses of action will be followed:

1. Data for the system level and for those equipments in the system not being changed or otherwise modified by the computer will be approximated. Although this may at first seem highly inaccurate, it must be realized that system equipments which

remain unchanged will have little or no impact on the relative comparisons in these analyses.

2. Alternatively, equipments not changed, modified, or otherwise impacted in a particular trade study could be left out of the LCC analysis. Care must be exercised, however, that all possible impacts have been considered. These may include the sharing of test equipment or the impact to an existing training course or technical manual.

All available data will be used, wherever applicable, to allow as complete and as concise a study as possible.

Section 3 SCIENTIFIC AND TECHNICAL INFORMATION

As part of the Phase I study, a literature search was performed on the open literature over the last 3 years, using key-words such as fault tolerance, error detection, error correction, self-checking, built-in-test, reconfiguration, etc. Approximately 400 papers, reports, books, and theses were located. If they were all studied in detail, a great many of these references would prove to be too vague or general to be very useful, or too theoretical to be applicable to real problems, or so engrossed in the details of a particular application to be of minimal value to other applications. However, after a cursory review, several studies of distributed fault-tolerant systems were identified that do not suffer from these defects and are related to this study. Some of these were cited in the statement of work.

While it is not necessary to summarize the results of all of these documents here, it is important to make a few comments about them. The comments given here are of two types:

1. Identification of new or key concepts or results which should be considered in the performance of this study.
2. Identification of limitations in the document that necessitate further analysis in attempting to apply the results to this study.

Specific references are given for each one.

3.1 ULTRASYSTEMS STUDY FOR AFAL

Reference: "Fault-Tolerant Avionics Systems Architectures Study" by Ultrasystems, Inc. AFAL-TR-74-102 (June 1974).

This is a general study of fault tolerance in distributed avionics systems, and was identified in the statement of work as being important background for this study.

Key Concepts:

1. The concept of selective redundancy is discussed and emphasized. Because redundancy is expensive, in terms of hardware, software, and performance, it is not cost-effective to blindly build in high fault tolerance to all elements of an avionics system. Rather, each subsystem must be analyzed for its own needs and merits, and a redundancy mechanism defined accordingly. This philosophy was designed into the ATAS application described in Section 2.1 above, and will be pursued in the Phase II evaluation.

2. A general methodology is outlined for the fault-tolerance analysis of a system. This methodology is the basis for the problem description in the statement of work.
3. Specific computation and communication requirements are given for selected subsystems.
4. A clear and useful description of a fault-tolerance benefits analysis is given.

Limitations:

1. No alternatives for the global bus approach are discussed.
2. There is no general discussion of the software recovery problem, and only a brief summary of a rollback mechanism.
3. Every building block is defined to be self-checking, and it is apparently assumed that this self-checking is adequate and dependable, and that no cooperative checking is either needed or useful.

3.2 UNIFIED DATA SYSTEM

References:

1. "The Unified Data System: A distributed Processing Network for Control and Data Handling on a Spacecraft", by David A. Rennels, et. al. NAECON 76.
2. "Software Techniques for a Distributed Real-Time Processing System", by Fred Lesh and Paul Lecoq. NAECON 76.
3. "Architectures for Fault-Tolerant Spacecraft Computers", by David A. Rennels. Proc. of IEEE (Oct. 1978).

This is a study of redundancy in spacecraft computers that require long-term (several years) unattended operation, and very low power, weight, and volume.

Key Concepts:

1. "Soft names" are used to address memory modules within a computing subsystem; i.e., the high-order address bits to which a given memory module will respond can be set by the program. This is a useful technique for reconfiguration of a failed memory module.
2. I/O is connected to a processor's main memory bus, and is accessed through the use of invalid memory addresses. This is called "memory mapped" I/O.
3. Very low data rates are needed (a few thousand bits per second).

Limitations:

1. The distributed system design is based in several key areas on unique characteristics of the spacecraft application; namely:
 - a. Message transmission delays of several milliseconds on the bus are acceptable.
 - b. Space experiments in the distributed subsystems can all be rigidly synchronized by a slow (2.5 ms) central system clock.
 - c. A recovery strategy that stops a faulty unit, switches in a back-up unit, reloads its memory, and restarts it, with an overall delay of several seconds, is satisfactory.

None of these approaches are expected to be acceptable in the ATAS application.

2. There is no discussion of alternative buses.
3. One high-level processor acts as a system executive, that needs to be duplexed to avoid a single-point failure.

3.3 DISTRIBUTED PROCESSOR/MEMORY SYSTEM

Reference: "Distributed Processor/Memory Architectures Design Program", by Texas Instruments, Inc. AFAL-TR-74-80 (Feb. 1975).

This is a more detailed hardware and software design study of an earlier Honeywell study of distributed processing in avionics systems. This appears to be the most complete and most applicable study available.

Key Concepts:

1. A two-level bus structure is defined; a local bus for subsystem internal communication, and a global bus for overall system control.
2. Associative matching of destination addresses for messages is used; i.e., a message on the global bus is addressed to a task, not to the processor that contains that task. This is especially useful for minimizing the software impact of re-configuration after a permanent failure. It also requires that the bus protocol has a basic broadcast capability.
3. There is a discussion of techniques for partitioning an application onto a distributed network (called "program construction"), by the use of directed graphs. This is an important aspect of any distributed processing application; one which is often ignored.

4. There are some comments about the applicability of MIL-STD-1553 to a distributed system bus. (These comments will be considered in the evaluation of 1553 during Phase II.)
5. Numerous bus-checking techniques are described, in some detail.
6. A system network simulator is described, although unfortunately not in very much detail. It provides the same functional capabilities as the performance model described in Section 2.5.1.
7. This study also emphasizes the importance of selective redundancy.
8. Bus traffic of 1 Mb/s is expected to be adequate, although there is no data supporting this figure. (This is consistent with the communications load for ATAS in Section 2.1.)

Limitations:

1. Each external I/O device is assumed privately connected to one of the processing elements. No consideration is given to allowing devices to be switchable from one processor to another, or perhaps to connecting the I/O devices via a bus.
2. There is a complete description of a unique processor architecture and design, although there is no rationale given for why this is necessary.
3. Each processor contains a local executive program, but there is also a global executive defined, that appears to execute in a dedicated processor. This is a potential single-point failure with no clear discussion of how to recover from its failure. Besides, many of its functions could reasonably be left to the application programs.
4. There is no discussion of software recovery techniques.
5. It is stated that physical reconfiguration (i.e., re-allocation of functions to recover from a failed hardware element) is not practical, but there is no discussion of why this conclusion is drawn.

3.4 WIDEBAND MULTIPLEX BUSES

Reference: "The Impact of Wideband Multiplex Concepts on Microprocessor - Based Avionic System Architectures", by Honeywell, Inc. AFAL-TR-78-4 (Feb. 1978)

This study is primarily addressing multiplex buses, and the potential usefulness of high bandwidth (such as expected from fiber optics). This is an intelligent and detailed review and evaluation of high-performance multiplex busing techniques.

Key Concepts:

1. Avionics systems don't require high bandwidth. A detailed analysis of the communications traffic for the A-7D avionics system was prepared, in the same way that the ATAS system was analyzed in Reference 3 and Section 2.1 above. The A-7D worst-case load was about 84 kb/s while the equivalent ATAS load was about 89 kb/s. However, the total bus bandwidth actually used up (including control words, redundancy information, message gaps, etc.) can be 10-30 times the raw system data rate. This is a major impact, if universally true.
2. Several existing multiplex buses as well as several proposed buses are described briefly and evaluated.
3. Three specific bus systems are proposed and evaluated for fiber optic technology. This is useful input for the fiber optic bus assessment to be included in Phase II.
4. The bus evaluation methodology in this report is basically the same as that employed in Phase I of this study, and described in Section 2.4 above; namely, a set of bus evaluation criteria were defined, a set of buses were described briefly based upon key characteristics, and then the buses were evaluated for each of the criteria. Of the 17 criteria defined in this reference, about 30% seem to be particularly oriented towards the high speed fiber-optic bus technology (and therefore not pertinent to the Phase I evaluation), and about 30% correlated fairly directly with criteria defined in Section 2.4. For the other criteria, the emphasis in Section 2.4 is on more detailed hardware functions, capabilities, and more detailed fault-tolerance mechanisms. The reference also included a set of weighting factors in the final evaluation matrix.

3.5 SOFTWARE IMPLEMENTED FAULT TOLERANCE

Reference: "SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control", by John H. Wensley, et. al. Proc. of the IEEE, (Oct. 1978).

This paper presents the results of a detailed study of the hardware and software design of a multiple-processor system, where redundancy and fault tolerance is achieved by software. It is not strictly a distributed system, since every processor can directly address (although only in read-only mode) the main store of every other processor. Hence there is no programmed inter-processor communication.

Key Concepts:

1. A given task is executed on several processors (under control of a global executive; not necessarily concurrently) and the outputs are voted on by software before they are passed on as inputs to other tasks. Each copy of an output value is obtained from a different processor over a different bus. Note that this technique effectively masks transient failures.
2. The global executive is treated as a task also, so it is executed in multiple and its outputs voted upon. Consequently, there is no problem of a designated processor to run it.
3. Selective redundancy is achieved by software scheduling; i.e., the more critical tasks can be executed three, four, or more times, if necessary.
4. A detailed software algorithm is given for synchronizing the clocks across the various processors.
5. Elimination of software faults is planned by proving the correctness of the application programs.
6. It is believed that ECC in the processor memory gives only slight improvement in processor reliability, and thus is not included. However, no justification is given for this conclusion.

Limitations:

1. It is assumed that all tasks are iterative; i.e., they are re-executed on a periodic basis. In an avionics system many tasks are, but some are not. It is not clear how non-iterative tasks are handled.
2. It is difficult to assess the total cost of this form of redundancy control, since part of it derives from the multiplicity of processors and buses, and part of it derives from the (selective) multiplicity of execution of tasks. One would expect the software overhead to be significant.
3. Communication on the bus is relatively slow ($\sim 10 \mu s$), since very little data movement is expected. However, there is no data given to support this conclusion.

3.6 ULTRASYSTEMS STUDY FOR NASA

Reference: "Definition and Trade-Off Study of Reconfigurable Airborne Digital Computer System Organizations", by Ultrasystems, Inc. NASA CR-132537 (Nov. 1974).

This is a study of redundant configurations, where the processors are all identical and all executing the same programs concurrently (and consequently are not operating as a distributed system).

Key Concepts:

1. Considerable discussion is given about how to measure the fault tolerance of a system. In particular, the notion of survivability is defined. In a reconfigurable system with redundant modules, this is the probability that some required minimal set of modules will continue to operate for the required mission time. It is one approach to considering the effects of transient failures, as well as imperfect processes of error detection, error isolation, and reconfiguration.
2. Software recovery by rollback is described, and the concept of rollahead introduced. (Unfortunately, the latter is only applicable to multiple executions of the same task).
3. A complementary analytic-simulative technique for calculation of predicted failure probabilities of multicomputer systems is described, and applied to a number of system configurations.
4. Some individual error-handling techniques are evaluated, such as NDRO memory, BIT, diagnostics, I/O wrap tests, voters, and reasonableness tests.

Limitations:

1. There are a number of very good and potentially useful concepts presented here, but the specific results are limited by the nondistributed characteristics of the configurations studied.
2. Some of the results are based upon computer load requirements for some specific applications, but that was proprietary data and not included in the report.

3.7 DRAPER LABORATORIES STUDY

Reference: "Digital Processing Analyses/Partitioning. Final Report", by Draper Laboratories. Report R-1122 (Nov. 1977).

This is a study of various multicomputer system configurations for digitally-guided weapon systems.

Key Concepts:

1. Application programs are described in structural architectural trees, which are used for the analysis of partitioning options.
2. General factors involved in computing life-cycle costs are defined, and various system configurations are evaluated in terms of those factors.
3. The bus traffic requirement for this set of applications is about 32 KB/s.

4. There is some discussion of the difficulties involved in using MIL-STD-1553A for busing.
5. Various bus configurations, bus protocols, and bus error-checking techniques are described and assessed.

Limitations: There are many relevant configuration and bus options discussed in this report, with general comparative comments. However, no technical detail is provided to support the evaluations.

3.8 DAIS FAULT-TOLERANCE STUDY

Reference: "A Fault-Tolerance Assessment of DAIS", by W. Heimerdinger and K. Fant, Honeywell. AFAL-TR-79-1007 (March 1979).

Key Concepts:

1. A labeled graph technique is used to describe the core system functions.
2. Specific types of messages and task interactions are described. These are consistent with previous IBM studies of distributed systems communications (Reference 7), except for the use of compools.
3. A fault catalog is developed for the current DAIS implementation, although this is functionally oriented rather than hardware oriented (as in the error catalog in Section 2.3 above).

Limitations:

1. Some general fault-tolerance mechanisms are described, but the only specific ones are directly related to the existing DAIS system bus.
2. The labeled graph technique is apparently highly thought of. However, the graphs in the report did not reproduce well and were hard to read, and it was not explained at all how the analysis of the graphs was done nor how conclusions about fault tolerance were derived therefrom. Consequently, it is difficult to assess the effectiveness of that particular technique.

3.9 STUDY FOR COMMERCIAL AIRCRAFT APPLICATION

Reference: "A Fault-Tolerant Multiprocessor Architecture for Aircraft", by Draper Labs. NASA Contractor Report 3010 (July 1978).

Key Concepts:

1. This is a dynamic TMR (triple-modular redundant) approach at the box level, to achieve less than 10^{-9} failures in the computer system per hour, in flights of up to 10 hours.
2. The concept of a bus guardian is described for implementing the switching function associated with reconfiguration.
3. A Markov model with 146 states is used for predicting system reliability.
4. There is considerable description of a software development methodology called higher order software (HOS), as an approach to reducing software life-cycle costs. It is not clear whether this is just a concept, or if it has in fact been used on a specific program.
5. A 14-processor hardware system was implemented and demonstrated, and it "illustrated all the significant aspects of the fault-tolerant architecture".

Limitations:

1. This is a tightly synchronized system with shared memory, rather than a distributed system.
2. The reliability objective is considerably higher than that needed for a tactical application.
3. It is claimed that programmed rollback is not an effective error handling technique, but there is no explanation or justification given for this statement.

3.10 AASMMA STUDY

Reference: "Advanced Avionics Systems for Multimission Applications (AASMMA)", by Boeing Co. Interim Technical Report No. 1 (Nov. 1978).

Key Concepts:

1. Three designs are given for an information-transfer system (i.e., system bus) for a distributed system; namely, a stationary master bus, a nonstationary master bus, and a contention bus. They are compared for bus efficiency, bus use, and trigger message detection speed. (These three buses are included in the bus evaluation in Section 2.4 above.) A coarse estimate is also given of BIU complexity.

2. Technology forecasts are given for microprocessors, memories, and BIUs. There is also considerable discussion of the impact of standardization (both hardware and software) on future system developments. (This will be useful input for the advanced technology assessment planned in Phase II.)
3. Several life-cycle cost models are compared, and a life-cycle cost model for software is described.

Limitations:

1. There is considerable analysis and data described for the three bus systems, but no conclusions or recommendations drawn.
2. The discussion is very closely geared towards how to modify DAIS for distributed processing.
3. A hierarchical bus structure is discussed, but with no rationale as to why this is good or important.

3.11 CARNEGIE - MELLON STUDIES IN FAULT TOLERANCE

References: "A Case Study of C.mmp, Cm*, and C.vmp: Part I - Experiences with Fault Tolerance in Multiprocessor Systems", by D. Siewiorek, et. al. Proc. IEEE, (Oct. 1978) p. 1178.

"A Case Study of C.mmp, Cm*, and C.vmp: Part II - Predicting and Calibrating Reliability of Multiprocessor Systems", by D. Siewiorek, et. al. Proc. IEEE (Oct. 1978), p. 1200.

These papers describe briefly three multiprocessor systems (called C.mmp, Cm*, and C.vmp) which have been custom designed and implemented at Carnegie-Mellon, and discuss various fault tolerance studies derived from them.

Key Concepts:

1. Many specific error-handling techniques are presented, such as the suspect/monitor model, the use of a local error register to support instruction retry, autodiagnosics, and dynamically trading reliability for performance. While a few are very specific to the existing hardware, many are of potential general interest and value.
2. Failure rate data is presented as measured for all three systems. (This data is difficult to compare with the failure rate data given in Section 2.2 above, partly because the functional breakdown of the systems are different, and partly because the systems in the reference are commercial hardware rather than MIL-SPEC hardware. However, in a few cases which could be compared, the failure rate data agrees

to within 50%.) Transient failures were found to occur an order of magnitude more frequently than solid failures. Most failures occur in main memory, and most of these are transient.

3. There is a clear discussion of system reliability modeling, and specific models are developed for all three systems. (The approach is the same as that discussed in Section 2.5.2 above.)

Limitations:

1. C.mmp and C.vmp are tightly coupled rather than distributed. In addition, the structure of Cm* appears distributed (local buses and global buses), but the address mapping device makes it look to the program like shared memory.
2. The failure rate data represents one of the very few sources of published data available on transient failures. However, the data is for commercial hardware rather than military hardware, and for processors from only one manufacturer (DEC).

3.12 MCF STUDY

Reference: "A Preliminary Study of Built-In-Test for the Military Computer Family (MCF)", by J. Clary, et. al. CORADCOM-76-0100-F (March 1979).

Key Concepts:

1. Fault tolerance features in a number of specific systems are summarized (JPL STAR, ESS 1A, AN/AYK-14, PDP-11/70 and PDP-11/60).
2. Failure rate data which were looked at show that 60% of the failure rate is in main memory, and 30% is in the CPU.
3. The MCF system is broken down into the same functional modules described in Section 2 above. Specific BIT suggestions are given and evaluated for each functional module.
4. One conclusion from the study is:
 - 80% failure detection can be achieved for MCF at a 10% hardware cost.
 - 80-90% can be achieved for 20% cost.
 - 90-95% can be achieved for 25% cost.

Limitations:

1. MCF as defined in this report is not a distributed system. However, most of the results are for single processors, and would still be of value.
2. One of the error factors considered in the study is false alarms; i.e., an indication of an error when no error is present. However, there is no discussion of how they can be distinguished from real transient failures, nor why it is necessary to do so.

3.13 INTEGRATED AVIONICS SYSTEM

Reference: "An Integrated Fault Tolerant Avionics System Concept for Advanced Aircraft", Draper Labs, Report R-1226 (February 1979).

This report documents a 3-month study performed for the Navy to establish a preliminary definition of an integrated fault and damage tolerant avionics architecture. It postulates a highly distributed system for use in aircraft in the mid-1990s, but with the Fault Tolerant Multiprocessor (FTMP) as a centralized control element. (FTMP is the same processing concept described in the Reference in Section 3.9, above).

Key concepts:

1. There is a major emphasis on selective redundancy, which is very important for avionics applications. A higher level of redundancy is required for flight critical functions than for noncritical functions.
2. There is a fairly detailed description of a modified 1553 protocol proposed as a dynamically reconfigurable bus. It is planned to interconnect 50-100 nodes, and its configuration is controlled by a highly reliable central computer. This bus appears to be quite a bit more general than is required in this study, both in terms of the number of terminals it is expected to interconnect, and the generality of the reconfigurability. However, it is a very good discussion of the problems associated with achieving a fault tolerant (and damage tolerant) bus.
3. There is a detailed discussion of the limitations inherent in MIL-STD-1553B for this type of application. These considerations will be directly applicable to the 1553B evaluation to be performed in Phase II of this study.
4. A general discussion of advanced technology trends into the 1990s is included. This perspective will be useful in the Phase II activity to evaluate advanced technology impacts on fault tolerance.

5. Processing requirements are described and estimated for a number of subsystem areas, including flight control, navigation, controls and displays, and communications. As with other such requirements definitions, it is difficult to directly correlate the functions listed in the report with those described in the ATAS system above (Section 2.1). The requirements for navigation and guidance are reasonably close (150 KOPS vs. 112 KOPS, and 18K words of storage vs. 15.6K words). This difference may reflect the executive program requirements. The JTIDS requirements are significantly different (1305 KOPS vs. 343 KOPS, and 92K words of storage vs. 38K words). This requirement is very dependent upon how the function is implemented; i.e., how much of the work is being performed by programmable processors rather than by direct hardware.
6. There is a discussion of highly reliable power supply distribution. This is an important consideration in a fault tolerant system, since the power supply can easily become a single-point failure source. However, this is not a general study of the problem; rather, it is an evaluation of one approach being made by the Navy to this problem (the Advanced Aircraft Electrical System).

Limitations

1. There is a general discussion of three generic approaches to fault tolerance; namely, independent primary and secondary resources, replicated primary resources, and resource pooling with dynamic allocation. After a number of evaluation criteria are considered, the pooled resource approach is recommended. However, alternative approaches to the pooled resource concept are not developed. Rather, FTMP is proposed as the only option to be considered, and this is used as the basis for the rest of the report.
2. Results of a reliability model are given, where the overall survival probability is divided into three phases; namely:
 - a. Probability of failure due to the lack of perfect coverage.
 - b. Probability of failure due to exhaustion of spares.
 - c. Probability of failure due to failures in bus controllers during reconfiguration.

However, the modeling in a and c is specifically oriented to the FTMP configuration, and does not appear to be directly applicable to this study.

Section 4

PHASE II STUDY PLAN

An intuitive description of the Phase II plan was given in Section 1, previously, along with an outline of the overall methodology of the complete study. This section presents the specific plan for carrying out the Phase II portion of the study. The plan presented here differs somewhat from that presented in the original proposal (Reference 2). This is the natural consequence of now having a better understanding of the problem and how best to address it. Therefore it is appropriate to first restate the problem as it is now viewed, and then to describe the proposed plan.

4.1 DEFINITION OF THE PROBLEM

The primary objective of the Phase II study is to evaluate the cost/effectiveness of fault tolerance techniques. To do this, it is first necessary to define more clearly what the fault tolerance techniques are.

Figure 4-1 is a flow diagram of a general procedure for responding to an error in the computer network. This procedure includes the following major steps:

1. The error must first be detected, by either hardware or software. No controlled response is possible for an error which is not detected.
2. If the function evidencing the error is retryable, and if the proper hardware is provided, the function can be retried. If successful, this converts a transient error into a nonerror, and the system can continue operation. If the retry is not successful, the error is considered to be solid.
3. Given that a solid error has occurred, the objective is to remove the failing element from the system. To do this it is necessary to identify the specific hardware element that is experiencing the failure; i.e., isolate the error to the functional module. (The functional modules are the blocks in the computer network shown in Figure 1-2). Note that this is not necessarily the same requirement as isolating to a field replaceable unit for maintenance purposes.
4. Once the failing module has been identified, a software procedure will analyse the existing configuration for spare modules, and will decide how to reconfigure the system to operate without the failing module. This would normally involve switching in a spare module in place of the bad one, if a spare is available.
5. With the spare module in place, the software will have to restructure some of its programs and data tables to ensure that the programs will now work with the new module, and no longer attempt to communicate with or control the bad module.

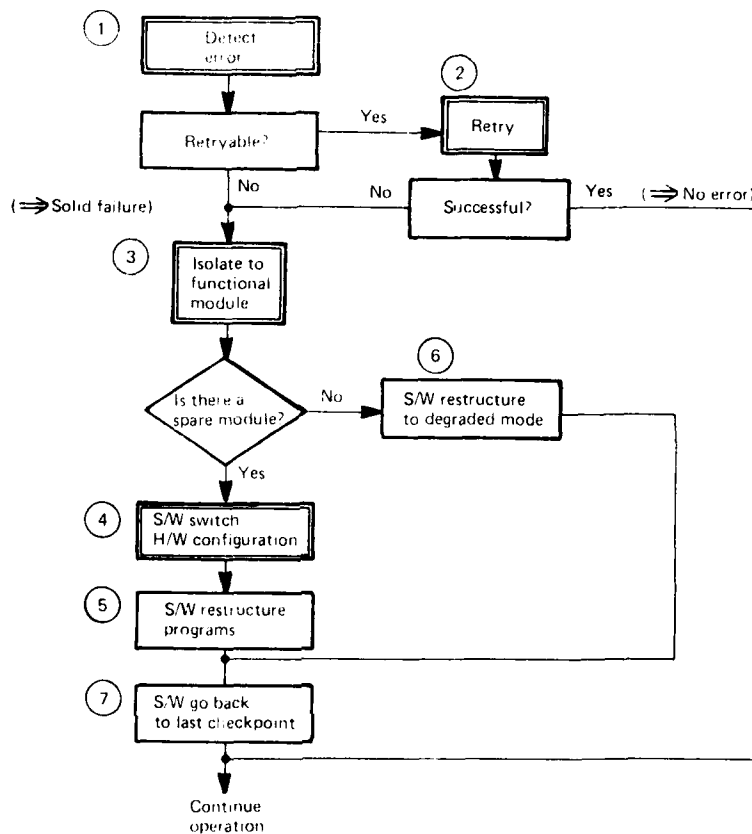


Figure 4-1. Error-Handling Procedure

6. If the failing functional module cannot be replaced (either because there are no spares, or there are no good spares left), the software may be able to restructure itself so that the system can still operate without the failed module. This would presumably result in operation in some form of degraded mode, with either reduced performance or reduced function, or both.
7. Performing the hardware reconfiguration and software restructuring is expected to take a nontrivial amount of real time. To restart the system operation, it will normally be necessary for the operational programs to go back to a data checkpoint, and restart from there.

In this way the system can continue operation in a presence of a solid failure. It could continue to operate after other failures, also, providing that there are either redundant modules or degraded modes available to match the specific failed modules as they occur. Ultimately, of course, these alternatives will be used up, and the whole system will fail.

In accordance with the statement of work, this study is focused on steps 1-4 in Figure 4-1. These are the hardware error handling facilities to be studied and evaluated, and the Phase II plan is directed to that end. Steps 5-7 are software functions, and are not addressed. (However, IBM has an IRAD program in place to investigate some of these problems, so that these software implications are not being ignored).

As a result of this breakdown of the problem, there are four general types of error handling techniques which need to be evaluated; namely, error detection, retry, isolation, and reconfiguration. It is important to identify these separately, since the evaluation method is different in each case.

The intent is to measure the cost and effectiveness of specific techniques that fall into each of these four classes. Table 4-1 summarizes the measures which need to be used to do this. The following observations will help clarify the information presented in the table:

1. Every hardware feature added for fault tolerance will have a cost, both in nonrecurring development and in recurring acquisition. The effect on life-cycle cost, however, needs to be determined.
2. Every increase in hardware will also increase the module failure rate, even though that increase may be small.
3. Some of the cost measures may be zero in specific cases. For example, many error detectors can be implemented in such a way that there is no performance loss. On the other hand some may have a significant performance penalty.
4. The cost measures listed are those that are expected to be permanent and inherent in the specific technique. For example, there is a performance impact whenever retry is invoked. However, this only happens when a retryable error occurs, and is reflected only in the total system recovery time. The performance loss intended in the chart is the performance loss that occurs continuously as a result of the retry circuits being included in the hardware design. Similarly, the hardware cost for reconfiguration is the cost of the built-in switching mechanisms, and not the cost of the spare modules used in a particular configuration.
5. The effectiveness measures for each class of error handling techniques are in fact different.

The important implication of this breakdown is that the four classes of techniques are relatively independent, and much of the evaluation can be done independently. Consider, for example, the class of retry techniques. In spite of all the error conditions and error detectors listed in the error catalog (Section 2.3), there are only a few retry mechanisms to be considered. The cost factors associated with these as well as effectiveness factors can be established independently from the other three classes of techniques.

TABLE 4-1. EVALUATION MEASURES

Error Handling Technique	Cost Measures	Effectiveness Measures
Error Detector	Hardware cost Failure rate increase Performance loss	Percent detection Coverage percent
Retry	Hardware cost Failure rate increase Software cost	Percent of transient errors Decrease in solid failure rate
Isolation	Hardware cost Failure rate increase Software cost	Probability of successful reconfiguration
Reconfiguration	Hardware cost of switching Failure rate increase Software cost	Probability of mission success

Ultimately, however, these evaluation factors must be combined to show the net effect on a distributed computer network. Figure 4-2 illustrates how these effects apply. An initial computer network configuration is defined, and descriptive parameters established. Error handling techniques from each of the four classes can be selected, and the parameters modified accordingly. The resulting data is then fed into the life-cycle cost model and the reliability model to determine system life-cycle cost and probability of mission success.

The Phase II plan reflects this approach to the evaluation of the error handling techniques.

There are two other principle objectives of the Phase II plan:

1. Perform a detailed evaluation of the buses selected in Section 2.4, and recommend the optimum choice for use in a fault-tolerant distributed computer network.
2. Consider the impact of new advanced technologies in both hardware and software, and estimate how they might affect the results of the technical evaluations.

These two objectives are addressed by separate tasks in the Phase II plan.

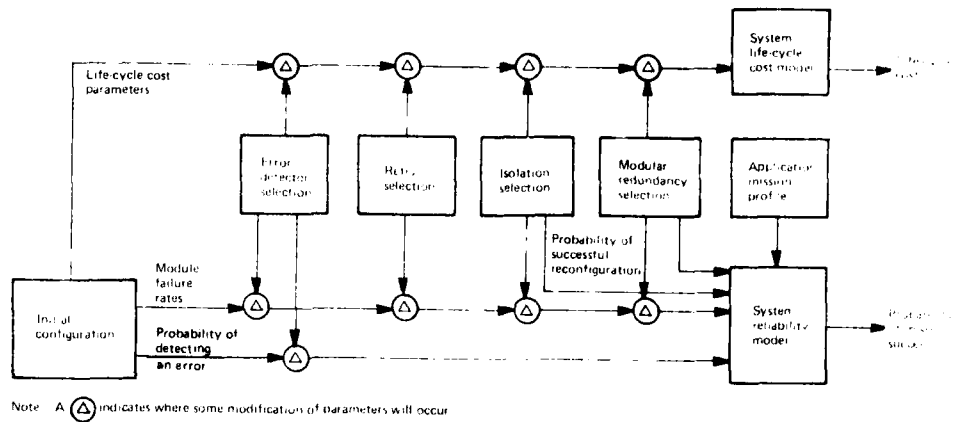


Figure 4-2. Combined System Evaluation

4-2 OUTLINE OF TECHNICAL EFFORT

The following sections describe a set of tasks to implement the Phase II study. This plan is defined so as to:

1. Meet the requirements of the statement of work
2. Fulfill the objectives described in Section 1 of this report
3. Implement the methodology as shown in Figure 1-1
4. Address the fault-tolerance evaluation in the form outlined in Section 4.1

Table 4-2 shows how the tasks described below relate to the steps in the methodology shown in Figure 1-1.

TABLE 4-2. APPLICABILITY OF PHASE II TASKS

Tasks	Building Block Characteristics	Define Objectives	Define Measures	Define & Measure Systems	Evaluate Results	Select Solution
2.1 Configuration Selection	X			X		
2.2 Fault Tolerance Selection	X	X	X	X		
2.3 Fault Tolerance Evaluation				X	X	
2.4 Communication Protocol	X			X	X	X
2.5 Conclusions						X

4.2.1 TASK 2.1 - CONFIGURATION ANALYSIS AND SELECTION

The objective of this task is to select a modest set of computer network configurations as bases for the evaluations in Task 2.3, and to provide detailed descriptive data about each.

The building blocks that will be considered for inclusion in a configuration are the processors, memories, and I/O given in Section 2-2, and the buses selected in Section 2-4. The number of configurations will be minimized, but with a reasonable cross-section of building block types. For example, the following comparisons should be covered by the configurations selected:

1. Larger processors vs. smaller processors
2. Symmetric (i.e., all processors identical) vs. nonsymmetric
3. Different bus/processor combinations.
4. I/O connected to processors, to an I/O bus, and to the system bus.

Each configuration will be reviewed for completeness by a systems engineer/designer. For example, a configuration containing microprocessors will need to specify a package with microprocessor chips and support chips to provide a workable interface to the computer network, and this is not available off-the-shelf today.

For each configuration, the ATAS application (from Section 2.1) will have to be partitioned to match the processor set, and the computational tasks allocated accordingly. This establishes the computing load, bus traffic, and I/O traffic to be supported by each processor.

Basic sizing, cost, and reliability parameters are given in Section 2-2 for the various building blocks, but they may need to be adjusted slightly to match the special design characteristics of the individual configurations.

The output of this task is a set of configurations to be analyzed. Each configuration description will include a block diagram of the configuration, with each block explicitly identified as to the processor type, BIU type, etc. The computing load, bus traffic, and I/O traffic for each computing element will be specified. Hardware size and failure rate for each block will be specified.

4.2.2 TASK 2.2 - FAULT TOLERANCE ANALYSIS AND SELECTION

There are two objectives of this task. One is to develop a detailed evaluation plan for the specific set of error-handling techniques that have been identified. The other is to update the descriptive parameters for each technique.

Some 41 unique error detectors were identified in the error catalog in Section 2.3. Several retry mechanisms are implied but not specified. There are also numerous redundant module options that can be included in each configuration (i.e., hot and cold spares). All these alternatives need to be explicitly spelled out, and then a plan developed for exactly how to evaluate them. It is clearly impractical to evaluate all possible combinations of error-handling techniques and configurations. The following simplifications need to be considered:

1. Some techniques might possibly be eliminated by inspection (i.e., engineering judgement).
 2. Some classes (see Table 4-1) may be evaluated independently.
 3. Some techniques may need to be evaluated individually, while others might be evaluated as a group.
 4. Some classes or techniques may not need to be evaluated at the system level.
 5. Some techniques may be inappropriate for certain configurations.
- The result of this effort will be an explicit plan of how the evaluations are to be done, and against which of the configurations defined in Task 2.1.

There are two areas where the data in the error catalog may be inadequate for the evaluation process. One is that the catalog does not include explicit retry procedures and reconfiguration switching options. These need to be explicitly spelled out, and cost factors developed. The other is that the descriptive data in the catalog is relatively general in nature. This data needs to be reviewed and possibly revised for each of the configurations selected in Task 2.1. For example, the cost figures in the error catalog are known to be technology and design dependent. Consequently, they may be different from one specific configuration to another.

Two outputs will result from this task. One is the detailed evaluation plan for the various error classes and error techniques, matched against the different test configurations. The other is a table of descriptive parameter data for all the error techniques, also matched against the evaluation plan.

4.2.3 TASK 2.3 - FAULT TOLERANCE EVALUATION

The overall objective of this task is to perform the evaluation of the fault tolerance techniques that have been identified and defined.

The specific evaluations to be done will be based upon the configurations defined in Task 2.1, and will use the data generated in Task 2.2, according to the evaluation plan also specified in Task 2.2. The evaluation process will follow three distinct steps:

1. For each test configuration (i.e., initial configuration modified with the set of fault tolerance techniques called for in the evaluation plan), the performance model (described in Section 2.5.1) must be run to ensure that the configuration meets the performance requirements spelled out for that configuration.
2. The four classes of error-handling techniques (as shown in Table 4-1) will be evaluated appropriately, as spelled out in the evaluation plan, and using the reliability modeling techniques discussed in Section 2.5.2.
3. The total system life-cycle cost and probability of mission success will be developed, using the models from Sections 2.5.3 and 2.5.2, respectively.

One important checkpoint identified for this task (in Section 4.3 below) is the complete analysis of the first configuration. It is expected that there will be many details to be worked out to ensure that all the necessary data exists for all the models, and to work out the flow of data between models.

The output of this task will be a large set of data points corresponding to the evaluation plan. This data will be used in drawing the conclusions in Task 2.5 below.

Note that it may be necessary to modify the evaluation plan as the task proceeds, in order to cover unusual or unexpected situations that may arise.

4.2.4 TASK 2.4 - COMMUNICATION PROTOCOL

The preliminary analysis of bus techniques was completed in Task 1.4, which resulted in the selection of a specific set of bus protocols to be studied in more depth in this task. The specific objectives of this task are the following:

1. Execute a detailed study of the selected bus protocols, and determine if there is one which is optimum for a distributed computer network in an avionics application.

2. Select one bus protocol (with its Bus Interface Unit) for inclusion in each candidate configuration identified in Task 2.1
3. Document the final communication protocol design.
4. Evaluate MIL-STD-1553B as to its applicability to fault tolerant distributed computing systems, as studied here.

There is one subtask covering each of these objectives.

4.2.4.1 Detailed Bus Analysis

A detailed analysis of the selected control procedures will be performed in this task. This task is equivalent to a detailed specification and analysis of the bus and the bus interface unit (BIU).

The BIU (Figure 4-3) has two distinct sections. The bus interface circuits section is dictated by the electrical characteristics of the bus and provides the data encoding/decoding, interface sequencing and clocking functions. The control section interfaces the processing element (PE) software to the bus interface section. The characteristics of the BIU are critical because they can limit the level of fault tolerance that can be achieved for the distributed computer network and they can have a major impact on software costs.

In general, the BIU must be able to provide the following capabilities:

1. Interface the PE software to the bus interface circuits such that the bus implementation, bus management or allocation scheme, and most of the communication protocol are transparent to the software.
2. Implement the required protocol at the bus interface.
3. Provide error detection and recovery for bus errors and redundancy management for bus failures.
4. Provide self-test and bus test capability that does not require PE software intervention.

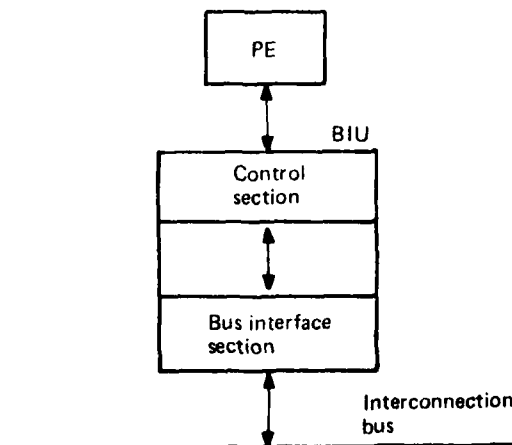


Figure 4-3. Sections of a Bus Interface Unit

5. Generate error status and provide failure location indications.

These factors must be included in each BIU analysis.

The major steps that are required in the performance of this subtask are:

1. Select an implementation for the communication control schemes selected in Task 1.4 which satisfy the above requirements. For these implementations evaluate the following parameters and estimate hardware costs and reliability:
 - a. maximum bus length
 - b. maximum number of devices
 - c. data encoding/decoding and checking scheme
 - d. bus access time, transfer rate, and transmission delays
2. Define the communication protocol:
 - a. method of initiating and terminating messages
 - b. message format
 - c. required handshaking
 - d. error response
 - e. self-test message formats
3. Estimate bus efficiency - Determine the number of bus cycles or transfers that are required to perform each communication type identified in Reference 7. Determine the same information for test messages.

These results will provide a relative measure of the efficiency of the communications schemes and their implementations. The efficiency figures will be used in the cost analysis task to measure bus utilization.

4.2.4.2 Candidate Selection

Based on preliminary assessments of bus life cycle costs and fault tolerance possibilities, one of the bus protocols will be selected for each candidate configuration to be evaluated. The specification of this protocol will include:

1. A top level summary of the bus protocol.
2. A functional description of the associated BIU, with cost and reliability data.
3. A definition of the error detection, isolation and recovery techniques which are inherent to the protocol.
4. Performance parameters associated with the operation of the bus.

This specification will be provided to Task 2.2, where additional fault tolerance techniques will be injected, leading to subsequent analysis and evaluation.

Feedback from Task 2.4 on system life cycle costs and fault tolerance effectiveness will be used to guide subsequent selections of candidate buses, and perhaps motivate changes to the bus (or BIU) design.

4.2.4.3 Documentation

The final configuration will be established as a result of the evaluations performed in Task 2.4. When documenting the final configuration, this subtask will provide a specification of the recommended bus protocol. This will specify the following:

1. Bus implementation
2. Bus allocation method
3. Functional description of the BIU, hardware cost and reliability estimates, and a description of the BIU software interface.
4. Error detection, isolation and recovery methods for transmission errors and BIU failures
5. Redundancy requirements and approach
6. Failure indications
7. Growth possibilities and limitations
8. Advantages and disadvantages

4.2.4.4 MIL-STD-1553B Evaluation

The MIL-STD-1553B standard bus is an important factor in future Air Force avionics system plans. However, there are a number of potential limitations on the use of MIL-STD-1553 in a fault tolerant distributed system. IBM studies of distributed systems have identified the following as possible problems:

1. Limitation of only 32 addressable elements
2. Messages are addressed to a hardware element, rather than to a software task.
3. The message size is limited to 32 words.
4. The bus requires a stationary master

The evaluation in Section 2.4 above indicates a number of other concerns. The MIL-STD-1553 bus will be evaluated based on the results of the overall system analysis and evaluation in Task 2.3 and will be compared with the bus protocol selected for the final configuration. This comparison will include cost, reliability, and fault tolerance factors. If possible, specific changes will be proposed to the MIL-STD-1553 specification to make it a cost effective solution for these types of systems.

1.2.5 TASK 2.5 - CONCLUSIONS AND RECOMMENDED SYSTEM

In Task 2.3, a detailed cost-effectiveness evaluation is performed on the various fault-tolerance techniques that were identified and defined. In this task, the totality of those evaluations will be assessed, and recommendations developed about each technique. It is anticipated that this recommendation will take one of three possible forms:

1. The technique is clearly cost-effective, and would be recommended for inclusion in all future avionics systems as a means of improving probability of mission success.
2. The technique is clearly not cost-effective (at least for the scope of present day technologies which are studied), and should not be recommended.
3. The technique provides a significant improvement in probability of mission success, but at a significant cost penalty. Its use would be recommended only in systems where higher reliability is an important requirement.

It is important to recognize this third possibility, because reliability is not a binary choice; rather, it is often a matter of "you get what you pay for".

In Task 2.4, four communication bus designs are studied and evaluation, and it is anticipated that one of these will be demonstrated superior to the other three.

In this task, a description of a distributed computer network will be developed, which is made up of the following:

1. One of the configurations defined in Task 2.1.
2. The bus and BIU selected from Task 2.4.
3. The fault-tolerance techniques as recommended above.

Cost, reliability, and performance data will be prepared for the individual modules, and for the system as a whole.

1.2.6 TASK 2.6 - ADVANCED TECHNOLOGY CONSIDERATIONS

The principal concern of the study is the quantification of fault tolerance attractiveness for state-of-the-art digital and avionics technology. This task addresses the probable courses of technology and evaluates their influence on the results of the study. Among the technology avenues to be pursued are:

1. Digital devices - scale, function, and reliability advances, especially VLSI and VHSIC.
2. Distributed system design methodology - limitations and projected improvements, especially parallelism.

- #### 4.2.7 FINAL REPORT

1. The set of configurations which were used for evaluation.
2. The error techniques and descriptive parameter data, as defined for each configuration.
3. The detailed evaluation plan, and the data obtained as a result of the evaluations.
4. The comparison data on the four candidate buses, and the detailed description of the recommended bus and its BIU.
5. The description of the final distributed computer network.
6. The results of the advanced technology considerations.

4.3 SCHEDULE

Figure 4-4 is the planned schedule and checkpoints for the Phase II study.

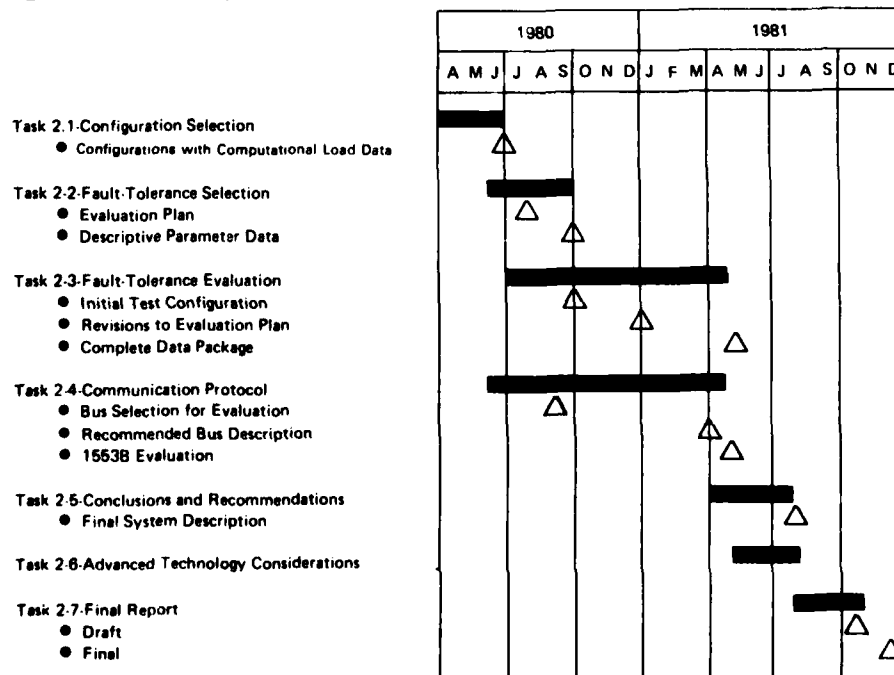


Figure 4-4. Phase II Schedule

Section 5

SUMMARY OF KEY RESULTS FROM PHASE I STUDY

Section 2 previously documented the technical results from the Phase I study. This section highlights a few key points and observations from this effort.

1. The bus loading on a distributed system in the advanced tactical fighter application is really not very high. In the worst case (i.e., if every major function was given its own computer) the bus traffic would be less than 6000 words/s, or approximately 93 kb/s. This is easily handled by a serial bus running at 1 MHz. If all the I/O traffic is also added to the same bus, this adds about 216 kb/s for a total of about 310 kb/s. The overhead associated with this data traffic, as well as that required for fault tolerance, needs to be looked at before determining whether or not a 1-MHz bus will be adequate.
2. Of the 12 computational functions for the ATAS application described in Section 1, 10 of them are well within the computational capability of existing microprocessors. The other two should be analyzed in more detail to determine if they can be further broken down.
3. The original notion of Task 1.5 was to develop incremental cost factors for each of the error detectors in the error catalog, that could be added to existing module values in Phase II. Unfortunately, the cost factors are very technology and design dependent, and will need to be re-assessed for each basic configuration to be used in Phase II.
4. The original plan was to use one generalized reliability model to establish the probability of mission success for all configurations and error-handling techniques to be evaluated in Phase II. As discussed in Section 2.5.2, this does not appear to be workable; rather, various modeling techniques will be required to handle the various levels of detail that need to be studied.

Section 6
REFERENCES

1. Statement of work, "Fault-Tolerant Computer Network" June 6, 1979. Solicitation No. F33615-79-R-1880.
2. IBM Proposal, "Fault-Tolerant Computer Network", July 6, 1979. IBM File No. 79-70N-022.
3. "Task 1. Computational Requirements for the Advanced Tactical Avionics System (ATAS)", by R.E. McNabb, M.H. Olson and J.B. Mac Pherson. Nov. 21, 1979. IBM Report No. 79-M56-007.
4. "A Systematic Approach to the Design of Digital Busing Structures", Thurber, Jensen, et. al. 1972 FJCC.
5. "Advanced Avionic Systems for Multimission Applications (AASMMA)", Interim Technical Report No. 1. Boeing Co. (Contract No. F 33615-77-C-1252).
6. Military Standard, "Aircraft Internal Time Division Command/Response Multiplex Data Bus - MIL-STD-1553B."
7. "Distributed Processing in Avionics Systems: A Communications Architecture", By W. T. Comfort, June 15, 1979. IBM Report No. 79-D68-002.
8. "Reliability Modeling and Analysis for Fault-Tolerant Computers", by Ying-Wah Ng, September 1976. UCLA-ENG-7698.
9. "Reliability Modeling and Prediction for Fault-Tolerant Digital Systems", by Ying-Wah Ng and Algirdas Avizienis, January 1979. UCLA Technical Report (Draft).

Appendix A PROTIME II MODEL SIMULATION METHODOLOGY

This section describes the equations used to perform the PROTIME II simulation. It is not meant to be a rigorous proof of the equations.

A.1 ASSUMPTIONS

The simulator is based on several assumptions which are reasonable for an environment in which resource requirements can be defined to the function level.

1. The I/O is distributed uniformly over the life of a function.
2. Within a function there is no overlap of I/O and CPU
3. The average MIPS rate is reasonable representative for each function.
4. Once a function obtains the memory it requires it will hold its memory until it completes.

With these assumptions in mind the following paragraphs discuss the simulation control and the required equations.

A.2 ANALYTIC EQUATIONS FOR RESOURCE CONSUMPTION

The total CPU time ($TCPU_f$) required by a function is given by:

$$TCPU_f = KOPS_f / MIPS_{fp} \quad (1)$$

where:

KOPS is the instruction count for the function f

MIPS_{fp} is the instruction execution rate of the function's processor.

The I/O time for a device used by the function ($TDEV_{f,d}$) is given by:

$$TDEV_{f,d} = (L_d + DR_d * DS_{f,d}) * DN_{f,d} \quad (2)$$

where:

L_d is the latency of device d

DR_d is the transfer rate of device d

$DS_{f,d}$ is the size of the transfer between the device d and the function

$DN_{f,d}$ is the number of transfers per execution of the function f.

The total I/O time for the function (TIO_f) is:

$$TIO_f = \sum_{d=1}^n TDEV_{f,d} \quad (3)$$

where n is the number of devices used by the function.

The minimum elapsed time for the function (assuming no resource contention) is:

$$TMIN_f = TCPU_f + TIO_f \quad (4)$$

The fractional part of the function's processor required by the function is:

$$CPUREQ_f = TCPU_f / TMIN_f \quad (5)$$

Likewise the fractional part of each device required by the function is:

$$DEVREQ_{d,f} = TDEV_{d,f} / TMIN_f \quad (6)$$

In actuality only the highest priority function can be assured of receiving its required CPU and device resources. The consumption of these resources for any general function may be limited by their availability. The fraction of the CPU resource required that may actually be consumed by the function is defined by:

$$\begin{aligned} CPUACT_f = 1 & \mid CPUREQ_f \leq CPUAVL_{fp} \\ & = CPUAVL_{fp} / CPUREQ_f \mid CPUREQ_f > CPUAVL_{fp} \end{aligned} \quad (7)$$

where: $CPUAVL_{fp}$ is the fraction of the function's processor available to the function after all higher priority functions have been serviced.

Similarly for each device used by the function

$$\begin{aligned} DEVACT_{f,d} = 1 & \mid DEVREQ_{f,d} \leq DEVAVL_d \\ & = DEVAVL_d / DEVREQ_{f,d} \mid DEVREQ_{f,d} > DEVAVL_d \end{aligned} \quad (8)$$

where: $DEVAVL_d$ is the fraction of each device available to the function after all higher priority functions have been serviced.

The total resources consumed (processor, devices, deviated) are in a fixed ratio so the actual consumption of resources for the function is given by:

$$\text{RATIO}_f = \min (\text{CPUACT}_f, \text{DEVACT}_{f,d}) \quad d = 1 \dots n \quad (9)$$

The fractional availability of CPU and device resource following the allocation to the function is:

$$\text{CPUAVL}_{fp} = \text{CPUAVL}_{fp} - \text{CPUREQ}_f * \text{RATIO}_f \quad (10)$$

$$\text{DEVLVL}_{f,d} = \text{DEVAVL}_{fd} - \text{DEVREQ}_{f,d} * \text{RATIO}_f \quad (11)$$

The time at which the function will complete at this level of resource consumption is given by:

$$\text{TCOMP}_f = \text{TREM}_f / \text{RATIO}_f \quad (12)$$

where: TREM_f is the time remaining for that function to complete (initially = TMIN_f).

These equations can be solved to calculate resource consumption each time the system state of the model changes. In reality equations 1 to 6 are solved only once (static parameters) at time zero. Equations 7 to 12 are solved each time the system state changes.

A.3 DISCRETE SIMULATION TIME CONTROL

Initially a timeline of system state changes is defined by a time ordered queue of external events. Each time the system state changes another event is generated at time:

$$t_{nf} = t_i + \min (\text{TCOMP}_f); \quad f = 1 \text{ to number of functions} \quad (13)$$

where: t_{nf} is the time of next most imminent function complete

t_i is the present value of simulation time.

The next time of system state change (t_{i+1}) will be:

$$t_{i+1} = \min (t_{nf}, t_{ex}) \quad (14)$$

where: t_{ex} is the time of occurrence of the external event at the head of the external event queue.

When all resource consumptions that define the t_i state of the system have been calculated, time t_{i+1} is calculated and the simulation time updated to that value.

A.4 ANALYTIC EQUATIONS FOR RESOURCE USE

At time t_{i+1} the resource use of the processors and devices can be calculated based on the resource consumption calculated at time t_i . The time interval t is:

$$\Delta t = t_{i+1} - t_i \quad (15)$$

The total CPU time ($TPROC_p$) up to t_{i+1} for a processor is:

$$TPROC_{pi+1} = TPROC_{pi} + \Delta t * (1 - CPUAVL_p) \quad (16)$$

and the total device time ($TOTDEV_d$) up to time t_{i+1} is:

$$TOTDEV_{di+1} = TOTDEV_{di} + \Delta t * (1 - DEVAVL_d) \quad (17)$$

The corresponding utilization's are then given by:

$$CPUUTIL_{pi+1} = TPROC_{pi+1} / t_{i+1} \quad (18)$$

$$DEVUTIL_{di+1} = TOTDEV_{di+1} / t_{i+1} \quad (19)$$

The only remaining calculation required to produce a closed loop is the calculation of time remaining for each function. This is given by:

$$TREM_{fi+1} = TREM_{fi} - RATIO_f * \Delta t \quad (20)$$

NOTE that in equation (12) $TREM_f$ should be subscripted by i . This was not done to avoid confusing the discussion at that point, which was prior to the consideration of time control.

A.5 MODEL EXECUTION

The set of equations presented above are used to determine resource consumption and use in PROTIME II. Equations (1) to (6) are solved only once at the beginning of each model run ($t=0$). Equations (7) to (17) and (20) are solved for each state of the system. After each time update is determined by (13) and (14), resource use is updated using equations (15) to (17). Equations (18) and (19) are used only for output.

The time remaining for each function is then updated by equation (20) and the resource consumption for the present system state is calculated using equations (7) to (12). Time is then advanced again thus forming the repeating loop used in PROTIME II.

Appendix B SAMPLE PROTIME II OUTPUT

```

*****00010000
*****00020000
*****00030000
*****00040000
*****00050000
*****00060000
*****00070000
*****00080000
*****00090000
*****00100000
*****00110000
*****00120000
*****00130000
*****00140000
*****00150000
*****00160000
*****00170000
*****00180000
*****00190000
*****00200000
*****00210000
*****00220000
*****00230000
*****00240000
*****00250000
*****00260000
*****00270000
*****00280000
*****00290000
*****00300000
*****00310000
*****00320000
*****00330000
*****00340000
*****00350000
*****00360000
*****00370000
*****00380000
*****00390000
*****00400000
*****00410000
*****00420000
*****00430000
*****00440000
*****00450000
*****00460000
*****00470000
*****00480000
*****00490000
*****00500000
*****00510000
*****00520000
*****00530000
*****00540000
*****00550000
*****00560000
*****00570000
*****00580000
*****00590000
*****00600000
*****00610000
*****00620000
*****00630000
*****00640000
*****00650000
*****00660000
*****00670000
*****00680000
*****00690000
*****00700000
*****00710000
*****00720000
*****00730000
*****00740000
*****00750000
*****00760000
*****00770000
*****00780000
*****00790000
*****00800000
*****00810000

```

INPUT SPECIFICATION FOR DPS MODEL

RUN TITLE = SYSTEM TEST ONE TEST 11
DATE =

(HH:MM:SS.TTT)
MAX SIM TIME = 00:01:40.000

SPECIFY PROCESSORS

NAME	KIPS (X.X)	CORE(K) (X.X)	ACPU NOT AVBL	COMMENTS
XXXXXXX	XXXXXX	XXXXXX	XX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PROCONE	100	200	63	SHOULD BE .625
PROCTWO	100	500	80	
PROC3	100	400		

SPECIFY DEVICES

NAME OR TYPE	QTY NO	LATENCY (X.X MS)	XFR RATE (X.X BPS)	COMMENTS
XXXXXXXX	XX	XXXXXXXX	XXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
DEV CE1	1	500.	1000.	
DEV CE2	1	500.	1000.	
DEV CE3		500.	1000.	
DEV CE4		500.	1000.	
DEV CE5		500.	1000.	
DEV CE6		500.	1000.	

SPECIFY EVENTS

(ALL)	INTERNAL EVENTS:	EXTERNAL EVENTS ONLY:		
	E T Y P E =	A N D I N G =	R E P E T P =	
EVENT NAME	X X X	A A A	R R R	HH:MM:SS.TTT
XXXXXXXX	X	A	R	00:00:00.000
EXEVEN1	X			5.
INEVEN1	1			

SPECIFY FUNCTIONS

FUNCTION NAME	STARTING EVT NAME	ENDING EVT NAME	SYS PRIO	INIT % DONE	COMMENTS
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXX	XX	XXXXXXXXXXXXXXXXXXXXXXXXXXXX
FUNC1	EXEVEN1	INEVEN1	1		
FUNC2	EXEVEN1	INEVEN1	2		
FUNC3	EXEVEN1	INEVEN1	2		
FUNC4	EXEVEN1	INEVEN1	1		
FUNC5	EXEVEN1	INEVEN1	1		

							00820000
							00830000
							00840000
							00850000
=6							00860000
							00870000
							00880000
							00890000
							*00900000
							*00910000
							*00920000
							*00930000
							*00940000
							*00950000
							*00960000
							*00970000
							*00980000
							00990000
							01000000
							01010000
FUNC1	PROCONE	10	200	DEVICE1	500	2	01020000
				DEVICE3	500	16	01030000
FUNC2	PROCONE	20	2000	DEVICE2	500	4	01040000
				DEVICE3	500	16	01050000
FUNC3	PROCTWO	30	200	DEVICE4	500	2	01060000
				DEVICE6	500	16	01070000
FUNC4	PROCTWO	40	1000	DEVICE5	500	2	01080000
				DEVICE6	500	8	01090000
FUNC5	PROC3	50	1600	DEVICE6	500	24	01100000
							01110000
							01120000
							01130000
							01140000
							01150000
							01160000
							01170000
							01180000

***ALL INPUT CARDS HAVE BEEN PROCESSED.

RUN TIME = 100.0000

PROCESSOR DATA

NAME	MEMORYSIZE	KIP RATE	MAXIMUM CPU
PROCONE	200.000	100.	0.370000
PROCTWO	300.000	100.	0.200000
PROC3	400.000	100.	1.000000

DEVICE DATA

NAME	LATENCY	TRANSFER RATE
DEVICE1	0.500000	1000.00
DEVICE2	0.500000	1000.00
DEVICE3	0.500000	1000.00
DEVICE4	0.500000	1000.00
DEVICE5	0.500000	1000.00
DEVICE6	0.500000	1000.00

EXTERNAL EVENT DATA

EXTERNAL EVENT NAME IS EXEVEN1 DEPENDENT EVENT	TYPE=	0	TIME OF OCCURRENCE=	5.0000	REPEAT TIME=	0.0
DEPENDENT FUNCTION						
FUNC1						
FUNC2						
FUNC3						
FUNC4						
FUNC5						

INTERNAL EVENT NAME IS INTERNAL EVENT
PRECEDENT EXTERNAL EVENT

PRECEDENT INTERNAL EVENT
PRECEDENT FUNCTION
PRECEDENT FUNCTION
PRECEDENT FUNCTION
PRECEDENT FUNCTION
PRECEDENT FUNCTION

NAME =FUNC1 PRIORITY = 1. MEMORY= 10.000 KOPS = 200.00
CPU REQUIRED =0.100000 CPU TIME = 2.00 IO TIME = 18.00 MINIMUM TIME = 20.00
DEVICE NAME DEVICE TIME PERCENT DEVICE REQUIRED
DEVICE1 2.00 0.100000
DEVICE3 16.00 0.800000
DEPENDENT EVENT
INEVENT 1
PRECEDENT EXTERNAL EVENT PRECEDENT INTERNAL EVENT PRECEDENT FUNCTION
EXEVENT1

NAME =FUNC2 PRIORITY = 2. MEMORY= 20.000 KOPS = 2000.00
CPU REQUIRED =0.500000 CPU TIME = 20.00 IO TIME = 20.00 MINIMUM TIME = 40.00
DEVICE NAME DEVICE TIME PERCENT DEVICE REQUIRED
DEVICE2 4.00 0.100000
DEVICE3 16.00 0.400000
DEPENDENT EVENT
INEVENT1
PRECEDENT EXTERNAL EVENT PRECEDENT INTERNAL EVENT PRECEDENT FUNCTION
EXEVENT1

NAME =FUNC3 PRIORITY = 2. MEMORY= 30.000 KOPS = 200.00
CPU REQUIRED =0.100000 CPU TIME = 2.00 IO TIME = 18.00 MINIMUM TIME = 20.00
DEVICE NAME DEVICE TIME PERCENT DEVICE REQUIRED
DEVICE4 2.00 0.100000
DEVICE6 16.00 0.800000
DEPENDENT EVENT
INEVENT1
PRECEDENT EXTERNAL EVENT PRECEDENT INTERNAL EVENT PRECEDENT FUNCTION
EXEVENT1

NAME =FUNC4 PRIORITY = 3. MEMORY= 40.000 KOPS = 1000.00
CPU REQUIRED =0.500000 CPU TIME = 10.00 IO TIME = 10.00 MINIMUM TIME = 20.00
DEVICE NAME DEVICE TIME PERCENT DEVICE REQUIRED
DEVICE5 2.00 0.100000
DEVICE6 8.00 0.400000
DEPENDENT EVENT
INEVENT1
PRECEDENT EXTERNAL EVENT PRECEDENT INTERNAL EVENT PRECEDENT FUNCTION
EXEVENT1

NAME =FUNC5 PRIORITY = 1. MEMORY= 50.000 KOPS = 1600.00
CPU REQUIRED =0.400000 CPU TIME = 16.00 IO TIME = 24.00 MINIMUM TIME = 40.00
DEVICE NAME DEVICE TIME PERCENT DEVICE REQUIRED
DEVICE6 24.00 0.600000
DEPENDENT EVENT
INEVENT1
PRECEDENT EXTERNAL EVENT PRECEDENT INTERNAL EVENT PRECEDENT FUNCTION
EXEVENT1

EXTERNAL EVENT QUEUE
1 EXEVENT1
2 ENDOFRUN
READY FUNCTION QUEUE

END OF RUN TIME = 100.0000

PROCESSOR DATA

NAME	MEMORY SIZE	MAX MEMORY	KIP RATE	CPU TIME	UTILIZATION	CPU AVAILABLE	MAXIMUM CPU
PROONE	200.000	30.000	100.000	85.00	0.850000	0.370000	0.370000
PROCTWO	500.000	70.000	100.000	92.00	0.920000	0.200000	0.200000
PROC3	400.00	50.000	100.000	16.00	0.160000	1.000000	1.000000

DEVICE DATA

NAME	LATENCY	TRANSFER RATE	DEVICE TIME	UTILIZATION	MAXIMUM RATE	RATE AVAILABLE
DEVICE1	0.50000000	1000.00	2.00	0.020000	0.100000	1.000000
DEVICE2	0.50000000	1000.00	4.00	0.040000	0.074000	1.000000
DEVICE3	0.50000000	1000.00	32.00	0.320000	1.000000	1.000000
DEVICE4	0.50000000	1000.00	2.00	0.020000	0.050000	1.000000
DEVICE5	0.50000000	1000.00	2.00	0.020000	0.040000	1.000000
DEVICE6	0.50000000	1000.00	48.00	0.480000	1.000000	1.000000

FUNCTION DATA

NAME =FUNC1	PROCESSOR =PROCON1	PRIORITY =	1.EXECUTIONS = 1
TIME REMAINING =	0.0	USAGE RATIO =1.000000	MINIMUM TIME = 20.00
READY TIME	START TIME	COMPLETE TIME	EXECUTION TIME
5.00	5.00	25.00	20.00
		WAIT TIME	RUNTIME
		0.0	20.00
NAME =FUNC2	PROCESSOR =PROCON2	PRIORITY =	2.EXECUTIONS = 1
TIME REMAINING =	0.00	USAGE RATIO =1.000000	MINIMUM TIME = 40.00
READY TIME	START TIME	COMPLETE TIME	EXECUTION TIME
5.00	5.00	65.54	60.54
		WAIT TIME	RUNTIME
		0.0	40.00
NAME =FUNC3	PROCESSOR =PROCTWO	PRIORITY =	2.EXECUTIONS = 1
TIME REMAINING =	0.00	USAGE RATIO =1.000000	MINIMUM TIME = 20.00
READY TIME	START TIME	COMPLETE TIME	EXECUTION TIME
5.00	5.00	45.00	40.00
		WAIT TIME	RUNTIME
		0.0	40.00
NAME =FUNC4	PROCESSOR =PROCTWO	PRIORITY =	3.EXECUTIONS = 1
TIME REMAINING =	0.00	USAGE RATIO =1.000000	MINIMUM TIME = 20.00
READY TIME	START TIME	COMPLETE TIME	EXECUTION TIME
5.00	5.00	95.00	90.00
		WAIT TIME	RUNTIME
		0.0	90.00
NAME =FUNC5	PROCESSOR =PROC3	PRIORITY =	1.EXECUTIONS = 1
TIME REMAINING =	0.0	USAGE RATIO =1.000000	MINIMUM TIME = 40.00
READY TIME	START TIME	COMPLETE TIME	EXECUTION TIME
5.00	5.00	45.00	40.00
		WAIT TIME	RUNTIME
		0.0	40.00

EXTERNAL EVENT DATA

EXTERNAL EVENT = EXEVENT 1

TIME OF OCCURRENCE
5.00

INTERNAL EVENT DATA

INTERNAL EVENT = INEVENT1

TIME OF OCCURRENCE
25.00

Appendix C
SAMPLE REPORTS FROM PROGRAM LCC

LIFE_CYCLE_COST_ANALYSIS_PROGRAM

2/ 2/1980

COST MULTIPLIER - 1.00

MTBF MULTIPLIER - 1.00

OPTION 1

STANDARD_ELEMENTS_FILE

<u>PARAMETER</u>	<u>VALUE</u>
SIE: ITEM ENTRY COST/NEW ITEM	450.00
SBR: BASE LABOR RATE/HOUR	20.00
SDR: DEPOT LABOR RATE/HOUR	30.00
SPSC: PACKAGING & SHIPPING COST/LB. - CONUS	0.60
SPSP: PACKAGING & SHIPPING COST/LB. - OVERSEAS	1.00
SID: INITIAL DATA MGT. COST/COPY/PAGE	0.0100
SIM: ITEM MGT. COST/ITEM/YEAR	230.00
SDM: DATA MGT. COST/PAGE/YEAR	6.00
SBMC: BASE MATERIAL CONSUMPTION RATE	2.50
SDMC: DEPOT MATERIAL CONSUMPTION RATE	6.50
DF: DISCOUNT FACTOR	0.10

OPTION 1, PG 2

LOGISTIC_FACTORS_FILE

<u>PARAMETER</u>	<u>VALUE</u>
NY: OPERATIONAL LIFE OF SYSTEM (YEARS)	20
NBC: NUMBER OF BASES - CONUS	28
NBO: NUMBER OF BASES - OVERSEAS	5
NSC: NUMBER OF SYSTEMS - CONUS	592
NSO: NUMBER OF SYSTEMS - OVERSEAS	50
NUMBER OF SYSTEMS AT EACH BASE - CONUS	
NUMBER OF BASES SYSTEMS AT BASE	
19 16	
9 32	
NUMBER OF SYSTEMS AT EACH BASE - OVERSEAS	
NUMBER OF BASES SYSTEMS AT BASE	
5 10	
SIN: COST/SYSTEM INSTALLATION	1000.00
OH: SYSTEM OPERATING HOURS/MONTH	60.0
NDS: NUMBER OF DEPOT WORK SHIFTS	2
RSTC: BASE RESUPPLY TIME - CONUS (HOURS)	288.
RSTO: BASE RESUPPLY TIME - OVERSEAS (HOURS)	360.
DMC: DEPOT REPLACEMENT CYCLE TIME (HOURS)	120.
DRC: DEPOT REPAIR CYCLE TIME (HOURS)	240.
BDSC: SHIPPING TIME TO DEPOT - CONUS (HOURS)	500.
BDSO: SHIPPING TIME TO DEPOT - OVERSEAS (HOURS)	600.
TAT: BASE TURNAROUND TIME (HOURS)	120.
WP: SYSTEM WARRANTY PERIOD (YEARS)	0.
AO1: SPARES OBJECTIVE (SYSTEM)	0.980
AO2: SPARES OBJECTIVE (SHOP)	0.980
DSSF: DEPOT STOCK SAFETY FACTOR	1.65

ACTIVATION_SCHEDULE

YEAR	MONTH											
	1	2	3	4	5	6	7	8	9	10	11	12
1	20	30	40	50	50	50	50	50	50	50	50	50
2	50	50	2	0	0	0	0	0	0	0	0	0

OPTION 1, PG 3

HARDWARE_DEFINITION_FILE

NUMBER OF REPLACEABLE UNITS = 10

LN	IN	NOMENCLATURE	NG	CRU	MTBF	UFP	W	FVS	RLS	RRS
1	1	SYSTEM	1	50000.	500.	0.000	80.0	0.0	4.0	0.0
2	2	LRU 1	1	22000.	1000.	0.300	40.0	2.0	10.0	3.0
3	3	SRU 1-1	1	3000.	10000.	0.000	2.0	0.0	5.0	5.0
4	3	SRU 1-2	4	3500.	6700.	0.000	2.0	0.0	5.0	5.0
5	3	SRU 1-3	2	2000.	10000.	0.000	2.0	0.0	5.0	5.0
6	3	SRU 1-4	1	1000.	10000.	0.000	8.0	0.0	5.0	5.0
7	2	LRU 2	2	10000.	2000.	0.200	12.0	1.0	8.0	3.0
8	3	SRU 2-1	3	1000.	8000.	0.000	2.0	0.0	5.0	5.0
9	3	SRU 2-2	1	1000.	8000.	0.000	4.0	1.0	5.0	5.0
10	2	LRU 3	2	4000.	2000.	0.200	3.0	1.0	7.0	3.0

LN	RMS	NRTS	COND	LV	LSEV	USEV	LR	USER				USER			
								1	2	3	4	1	2	3	4
1	20.	0.900	0.000	0	0	0.0	0	0	0	0	0	0.0	0.0	0.0	0.0
2	200.	0.080	0.001	1	1	3.0	1	2	0	0	0	3.0	0.0	0.0	0.0
3	50.	0.990	0.010	2	0	0.0	2	3	0	0	0	2.0	0.0	0.0	0.0
4	50.	0.990	0.010	2	0	0.0	2	3	0	0	0	2.0	0.0	0.0	0.0
5	50.	0.990	0.010	2	0	0.0	2	3	0	0	0	2.0	0.0	0.0	0.0
6	50.	0.990	0.010	2	0	0.0	2	3	0	0	0	2.0	0.0	0.0	0.0
7	200.	0.080	0.001	1	1	2.0	1	2	0	0	0	2.0	0.0	0.0	0.0
8	50.	0.990	0.010	2	0	0.0	2	3	0	0	0	2.0	0.0	0.0	0.0
9	50.	0.990	0.001	2	0	0.0	2	3	0	0	0	2.0	0.0	0.0	0.0
10	75.	0.200	0.020	1	0	0.0	1	2	0	0	0	3.0	0.0	0.0	0.0

OPTION 1, PG 4

SUPPORT EQUIPMENT DEFINITION FILE

NUMBER OF LINE ITEMS OF SUPPORT EQUIPMENT = 3

<u>LINE NUMBER</u>	<u>NOMENCLATURE</u>	<u>COST (CSE)</u>	<u>O&M COST FACTOR (COM)</u>
1	VERIFICATION SET	20,000.	0.010
2	LRU TEST SET	50,000.	0.010
3	SRU TEST SET	100,000.	0.010

CONTRACTOR DATA FILE

<u>PARAMETER</u>	<u>VALUE</u>
ACS: ACQUISITION COST/SYSTEM	50,000.
BTC: BASE LEVEL TRAINING COST	20,000.
DTC: DEPOT LEVEL TRAINING COST	150,000.
DCB: DATA ACQUISITION COST (BASE LEVEL MANUALS)	50,000.
DCD: DATA ACQUISITION COST (DEPOT LEVEL MANUALS)	20,000.
DCO: DATA ACQUISITION COST (OTHER)	30,000.
NPB: PAGES OF DATA (BASE LEVEL MANUALS)	150
NPD: PAGES OF DATA (DEPOT LEVEL MANUALS)	900
NPO: PAGES OF DATA (OTHER)	150
NI: NUMBER OF NEW INVENTORY ITEMS	300
CRSC: CONTRACTOR BASE RESUPPLY TIME - CONUS	360.
CRSD: CONTRACTOR BASE RESUPPLY TIME - OVERSEAS	480.
CIMC: CONTRACTOR REPAIR CYCLE TIME	200.
WFP: WARRANTY PRICE	0.

RELIABILITY GROWTH PROFILE

1.1	1.2	1.3	1.4	1.5	1.6	1.6	1.6	1.6	1.6	1.6	1.6
1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6

OPTION 2

SUPPORT EQUIPMENT REQUIREMENTS

LINE ITEMS OF SUPPORT EQUIPMENT REQUIRED AT BASE LEVEL

<u>EQUIPMENT</u>	<u>QUANTITY</u>	<u>COST</u>
VERIFICATION SET	33	660,000.
LRU TEST SET	33	1,650,000.

LINE ITEMS OF SUPPORT EQUIPMENT REQUIRED AT DEPOT LEVEL

<u>EQUIPMENT</u>	<u>QUANTITY</u>	<u>COST</u>
LRU TEST SET	1	50,000.
SRU TEST SET	1	100,000.

OPTION 3

SPARES REQUIREMENTS (DETAILED)

DEPOT LEVEL SPARES REQUIREMENTS

<u>REPLACEABLE UNIT</u>	<u>SPARES</u>	<u>COST</u>
LRU 1	5	110,000.
SRU 1-1	5	15,000.
SRU 1-2	21	73,500.
SRU 1-3	9	18,000.
SRU 1-4	5	5,000.
LRU 2	5	50,000.
SRU 2-1	14	14,000.
SRU 2-2	6	6,000.
LRU 3	9	36,000.

BASE LEVEL SPARES REQUIREMENTS (CONUS)

SYSTEMS/

BASE

BASES

16

19

LRU SPARES REQUIREMENTS

<u>REPLACEABLE UNIT</u>	<u>SPARES/BASE</u>	<u>TOTAL</u>	<u>COST</u>
LRU 1	0	0	0.
LRU 2	1	19	190,000.
LRU 3	1	19	76,000.

SRU SPARES REQUIREMENTS

<u>REPLACEABLE UNIT</u>	<u>SPARES/BASE</u>	<u>TOTAL</u>	<u>COST</u>
SRU 1-1	0	0	0.
SRU 1-2	1	19	66,500.
SRU 1-3	0	0	0.
SRU 1-4	0	0	0.
SRU 2-1	1	19	19,000.
SRU 2-2	0	0	0.

32

9

LRU SPARES REQUIREMENTS

<u>REPLACEABLE UNIT</u>	<u>SPARES/BASE</u>	<u>TOTAL</u>	<u>COST</u>
LRU 1	0	0	0.
LRU 2	1	9	90,000.
LRU 3	1	9	36,000.

SRU SPARES REQUIREMENTS

<u>REPLACEABLE UNIT</u>	<u>SPARES/BASE</u>	<u>TOTAL</u>	<u>COST</u>
SRU 1-1	0	0	0.
SRU 1-2	1	9	31,500.
SRU 1-3	0	0	0.
SRU 1-4	0	0	0.
SRU 2-1	1	9	9,000.
SRU 2-2	0	0	0.

OPTION 3, PG 2

BASE LEVEL SPARES REQUIREMENTS (OVERSEAS)

SYSTEMS/

BASE

BASES

10

5

LRU SPARES REQUIREMENTS

<u>REPLACEABLE UNIT</u>	<u>SPARES/BASE</u>	<u>TOTAL</u>	<u>COST</u>
LRU 1	0	0	0.
LRU 2	1	5	50,000.
LRU 3	1	5	20,000.

SRU SPARES REQUIREMENTS

<u>REPLACEABLE UNIT</u>	<u>SPARES/BASE</u>	<u>TOTAL</u>	<u>COST</u>
SRU 1-1	0	0	0.
SRU 1-2	1	5	17,500.
SRU 1-3	0	0	0.
SRU 1-4	0	0	0.
SRU 2-1	1	5	5,000.
SRU 2-2	0	0	0.

CONDEMNATION SPARES REQUIREMENTS

<u>REPLACEABLE UNIT</u>	<u>SPARES</u>	<u>COST</u>
LRU 1	6	132,000.
SRU 1-1	6	18,000.
SRU 1-2	36	126,000.
SRU 1-3	12	24,000.
SRU 1-4	6	6,000.
LRU 2	6	60,000.
SRU 2-1	23	23,000.
SRU 2-2	1	1,000.
LRU 3	120	480,000.

OPTION 4

SPARES REQUIREMENTS (UNIT TOTALS)

<u>REPLACEABLE UNIT</u>	<u>SPARES</u>				<u>TOTAL COST</u>
	<u>DEPOT</u>	<u>BASE</u>	<u>CONDEMNATION</u>	<u>TOTAL</u>	
LRU 1	5	0	6	11	242,000.
SRU 1-1	5	0	6	11	33,000.
SRU 1-2	21	33	36	90	315,000.
SRU 1-3	9	0	12	21	42,000.
SRU 1-4	5	0	6	11	11,000.
LRU 2	5	33	6	44	440,000.
SRU 2-1	14	33	23	70	70,000.
SRU 2-2	6	0	1	7	7,000.
LRU 3	9	33	120	162	648,000.

OPTION 5

MANPOWER REQUIREMENTS
(MANHOURS PER YEAR)

<u>YEAR</u>	<u>FLIGHT LINE</u>	<u>BASE</u>	<u>DEPOT</u>
1	2330.	2378.	1921.
2	5045.	5148.	4159.
3	4672.	4767.	3852.
4	4325.	4413.	3565.
5	4025.	4107.	3318.
6	3765.	3841.	3104.
7	3636.	3710.	2997.
8	3636.	3710.	2997.
9	3636.	3710.	2997.
10	3636.	3710.	2997.
11	3636.	3710.	2997.
12	3636.	3710.	2997.
13	3636.	3710.	2997.
14	3636.	3710.	2997.
15	3636.	3710.	2997.
16	3636.	3710.	2997.
17	3636.	3710.	2997.
18	3636.	3710.	2997.
19	3636.	3710.	2997.
20	3636.	3710.	2997.

OPTION 6

TOTAL COST SUMMARY (BY CATEGORY)

	UNDISCOUNTED COST	PRESENT VALUE COST
INITIAL TRAINING	170,000.	170,000.
DATA ACQUISITION	100,000.	100,000.
ITEM ENTRY	135,000.	135,000.
DATA MANAGEMENT	8,556.	8,556.
PRIME HARDWARE	32,100,000.	31,636,364.
SUPPORT EQUIPMENT	2,460,000.	2,424,469.
INITIAL SPARES	1,808,000.	1,781,886.
INSTALLATION	642,000.	632,727.
TOTAL ACQUISITION COST	37,423,556.	36,889,002.
FLIGHT LINE MAINT.	1,525,177.	726,267.
BASE LEVEL MAINT.	1,643,526.	782,623.
DEPOT LEVEL MAINT.	2,976,137.	1,417,193.
ITEM MANAGEMENT	1,380,000.	646,179.
DATA MANAGEMENT	144,000.	67,427.
PACKING & SHIPPING	65,200.	31,047.
S.B.MAINTENANCE	491,116.	229,494.
TOTAL O&M COST	8,225,157.	3,900,233.
TOTAL LIFE CYCLE COST	45,648,713.	40,789,235.

OPTION 7

TOTAL COST SUMMARY (BY YEAR)

SYSTEM OPERATIONAL LIFE = 20 YEARS

<u>YEAR OF</u> <u>PROGRAM</u>	<u>UNDISCOUNTED</u> <u>COST</u>	<u>PRESENT VALUE</u> <u>COST</u>
1	31,836,156.	31,836,156.
2	6,398,227.	5,816,570.
3	487,317.	402,741.
4	458,570.	344,531.
5	433,805.	296,295.
6	412,247.	255,973.
7	401,599.	226,692.
8	401,599.	206,084.
9	401,599.	187,349.
10	401,599.	170,317.
11	401,599.	154,834.
12	401,599.	140,758.
13	401,599.	127,962.
14	401,599.	116,329.
15	401,599.	105,754.
16	401,599.	96,140.
17	401,599.	87,400.
18	401,599.	79,454.
19	401,599.	72,231.
20	401,599.	65,665.
-----	-----	-----
TOTAL	45,648,713.	40,789,235.

CYCLE 1 VARIABLE - ACS SET TO

3000.0000

OPTION 8

SENSITIVITY ANALYSIS

SENSITIVITY PARAMETER -----ACS-----	UNDISCOUNTED TOTAL COST	PRESENT VALUE TOTAL COST
3000.000	15,474,713.	11,051,053.
4000.000	16,116,713.	11,683,780.
5000.000	16,758,713.	12,316,507.
6000.000	17,400,713.	12,949,235.
7000.000	18,042,713.	13,581,962.